VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Siru Saranpää

LINUX-PALVELIMEN ASENNUS JA KONFIGUROINTI

Liiketalous
2022

VAASAN AMMATTIKORKEAKOULU
Liiketalous

TIIVISTELMÄ

Tekijä	Siru Saranpää
Opinnäytetyön nimi	Linux-palvelimen asennus ja konfigurointi
Vuosi	2022
Kieli	suomi
Sivumäärä	50
Ohjaaja	Antti Mäkitalo

Opinnäytetyössä käsitellään Linux-pohjaisen palvelimen asennusta ja konfigurointia keskittyen erityisesti niihin seikkoihin, joita tulee ottaa huomioon palvelinkäytössä. Lisäksi kuvaillaan uuden analytiikkapalvelimen asennusta sekä siinä tehtyjä konfigurointeja.

Teoreettisena viitekehyksenä on Linux-järjestelmä ja sen toiminta. Keskeisinä käsitteinä on erinäisiä Linux-järjestelmän termejä ja komentoja. Tutkimusaineistona käytettiin kirjallisuutta sekä verkkolähteitä.

Analytiikkapalvelimen asennuksessa palvelimelle asennettiin ja konfiguroitiin CUDA, OpenCV, Nginx, Nagios, SSHFS ja SSH-tunnelointi sekä PostgreSQL-palvelin. Näiden palvelinten ja ohjelmistojen konfiguroinnissa onnistuttiin. Lopputuloksena voitiin päätellä, että Linux on kevyt, turvallinen ja helppokäyttöinen käyttöjärjestelmä palvelinkäyttöön, jota voidaan käyttää sekä yksinkertaisiin että monimutkaisiin käyttötarkoituksiin.

Avainsanat linux, palvelin, projekti, tietoturva

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Liiketalous

ABSTRACT

Author	Siru Saranpää
Title	The Installation and Configuration of a Linux-Based Server
Year	2022
Language	Finnish
Pages	50
Name of Supervisor	Antti Mäkitalo

The thesis addresses the installation and configuration of a Linux-based server, focusing especially on the matters that are important to consider in server use. It also describes the installation of a new analytics server and the configurations that were done. The aim was to create a server that could be used for other students' machine learning tasks and for development.

The central terms that were used are terms of the Linux system and the names of commands. Books and electronic materials were used for information.

The analytics server had CUDA, OpenCV, Nginx, Nagios, SSHFS and SSH tunneling, and PostgreSQL installed on it. The installation and configuration of these servers succeeded. In the end it could be determined that Linux is a lightweight, safe, and easy operating system for server use that can be used for either simple or advanced purposes.

Keywords linux, server, project, information security

SISÄLLYS

KUVIOLUETTELO

1 JOHDANTO

Tässä opinnäytetyössä tulen käsittelemään Linux-pohjaisen palvelimen asennusta ja konfigurointia. Pyrin kertomaan kaikista niistä osa-alueista, jotka ovat tärkeimpiä stabiilin ja turvallisen Linux-palvelinympäristön luonnissa. Kerron myös, kuinka hyödynsin näitä tietoja saamassani toimeksiannossa. Linux on hyvin vahvasti esillä palvelinmaailmassa sen yksinkertaisuuden, stabiiliuden ja turvallisuuden vuoksi. Avoin lähdekoodi mahdollistaa myös järjestelmän muokkaamisen minkälaiseksi tahansa. Ei ole siis mikään ihme, että todella monet isot IT-yritykset ovat valinneet Linuxin palvelinkäyttöjärjestelmäkseen.

1.1 Miksi Linux?

W3Techsin päivittäin päivittyvän palvelinkäyttöjärjestelmävertailun mukaan Linux- tai Unix-pohjaisia palvelimia on tällä hetkellä jopa 78,7 prosenttia kaikista heidän testaamistaan palvelimista, joista käyttöjärjestelmä pystytään päättelemään. (W3Techs 2021) Flaunt Digitalin mukaan Linuxin etuja palvelinkäytössä on sen stabiliteetti ja valinnanvara, joka tarkoittaa, että koska Linuxista on valtavasti erilaisia versioita, voidaan niistä valita omaan käyttöön sopivin, jolloin kaikesta ylimääräisestä ja tarpeettomasta päästään eroon. Muita etuja ovat muun muassa etusija pilvipalveluissa, lähdekoodin avoimuus, turvallisuus sekä edullisuus. Lähdekoodin avoimuuden ansiosta esimerkiksi uusien ohjelmistojen asennus on Linux-palvelimissa helppoa, ja ohjelmistot integroituvat keskenään paremmin. Avoin lähdekoodi parantaa myös turvallisuutta, sillä mahdolliset viat ja haavoittuvuudet voidaan löytää sen avulla nopeammin. (Flaunt Digital 2018)

1.2 Linux vai Windows?

Toinen suosittu palvelinkäytössä oleva käyttöjärjestelmä on Microsoftin Windows Server. Yleinen kysymys onkin, kannattaako palvelinkäyttöön valita Linux vai Windows. Windows-palvelimien eduiksi mainitaan aloittelijaystävällisyys graafisen käyttöliittymän ansiosta, huoltotuki, Windows-ohjelmistotuki, sekä automaattiset

päivitykset. Huonoja puolia taas ovat isommat kustannukset lisenssien takia, isompi haavoittuvuus haittaohjelmille sekä sen isommat laitteistovaatimukset. (phoenixNAP 2021) Riippuukin siis palvelimen tarkoituksesta, kumpi käyttöjärjestelmä sopii siihen paremmin. Iso asia Windows-palvelimissa on sen Active Directory-ominaisuus. Active Directoryn avulla voidaan tallentaa tietoa käyttäjätunnuksista sekä niiden oikeuksista. Näin voidaan varmistua, että kirjautumista yrittävä henkilö on varmasti aito. Active Directoryn avulla voidaan myös synkronoida käyttäjien tietoja useammille laitteille, jolloin esimerkiksi käyttäjän vaihdettua salasanansa se päivittyy automaattisesti myös muille Active Directoryyn liitetyille laitteille. Active Directoryä käytetään yleensä yrityksissä ja organisaatioissa eri laitteiden yhdistämiseen. (Quest 2022) Tämän takia Windows-palvelin voi olla haluttu isoissa yrityksissä ja organisaatioissa, joissa on monia eri laitteita, jotka käyttävät Windows-käyttöjärjestelmää.

2 ANALYTIIKKAPALVELIMEN TAUSTA, TARKOITUS JA TAVOITTEET

Opinnäytetyön projektin taustana on saamani toimeksianto Wikimedia Suomi ry:ltä koskien heidän tämänhetkistä projektiaan Helsinki Rephotographya. Projekti tehdään yhteistyössä Wikimedia ry:n kumppanin Ajapaikin kanssa, joka on Viron valokuvaperintöyhdistyksen voittoa tavoittelematon joukkoistussivusto.

2.1 Tausta ja tarkoitus

Ajapaikin tarkoituksena on säilyttää historiallisia kuvia sekä antaa käyttäjille mahdollisuus uudelleenvalokuvata niitä. Näin pystytään näkemään, miten jokin paikka on aikojen saatossa muuttunut ja kehittynyt. Helsinki Rephotography keskittyykin nimensä mukaisesti Helsingin alueen uudelleenvalokuvaukseen. Projektia tehtäessä kuitenkin huomattiin, että Ajapaikin pääpalvelimella ei ole tarpeeksi levytilaa ja tehoa tehdä joitakin vaativaa laskentaa tarvitsevia tehtäviä. Tästä syystä päätettiin asentaa uusi palvelin, jonka tarkoituksena olisi toimia kehitysalustana raskaampaan laskemiseen, jota ei voida tehdä pääpalvelimella. Haluttiin myös, että palvelin voisi toimia alustana toisten opiskelijoiden koneoppimiseen liittyvissä opinnäytetöissä.

2.2 Tavoitteet

Projektin tavoitteena on asentaa ja dokumentoida uusi Linux-pohjainen palvelin. Ensisijaisesti suositaan avoimen lähdekoodin ohjelmistoja. Aivan ensiksi palvelimelle asennetaan Linux-käyttöjärjestelmä ja siihen SSH-etähallintaohjelmisto. SSH eli Secure Shell on avoimen lähdekoodin etähallintaohjelmisto ja protokolla, joka mahdollistaa kahden tietokoneen kommunikoinnin keskenään tietoturvallisesti. Tämän jälkeen asennetaan monitorointiohjelmisto, johon valittiin Nagios. Nagios on avoimen lähdekoodin monitorointiohjelmisto, jonka avulla voidaan monitoroida palvelimen eri järjestelmiä ja lähettää hälytyksiä, mikäli jossain järjestelmässä havaitaan ongelmia. (Nagios 2021) Nagiosia muokataan myös niin, että se osaa lähettää hälytykset Ajapaikin Slack-kanavalle.

Koneoppimista varten tarvittiin myös OpenCV-ohjelmisto ja NVIDIA:n CUDA-kirjasto. OpenCV on avoimen lähdekoodin konenäkökirjasto, joka sisältää yli 2500 algoritmia, joiden avulla tietokone pystyy tunnistamaan muun muassa kasvoja ja esineitä. Kirjasto on käytössä myös isoilla IT-yrityksillä, kuten Googlella, Yahoolla ja Microsoftilla. (OpenCV, 2021) NVIDIA:n CUDA-kirjasto taas mahdollistaa NVIDIA:n näytönohjainten käytön OpenCV:llä, mikä on pakollista vaativiin koneoppimistehtäviin, joiden tekemisessä pelkällä suorittimella menisi hyvin kauan.

Palvelimelle haluttiin myös peilata Ajapaikin nykyinen tietokanta sekä kuvat. Tätä varten palvelimelle asennetaan PostgreSQL-palvelin, jota Ajapaik käyttää pääpalvelimellaan. PostgreSQL on ilmainen, avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä, joka korostaa laajennettavuutta sekä SQL-määräystenmukaisuutta. (PostgreSQL, 2021) Ajapaik käyttää PostgreSQL:ää esimerkiksi suositun MariaDB:n sijasta, sillä PostgreSQL:ssä on ensiluokkaiset GIS-ominaisuudet. GIS-ominaisuuksilla (Geographic Information System) tietokantaan voidaan tallentaa maantieteellistä dataa (Wiki.GIS 2022). Tälle PostgreSQL-palvelimelle peilattaisiin Ajapaikin tietokanta. Kuvat ja muut tiedostot voitaisiin peilata rsync-ohjelmalla, joka on avoimen lähdekoodin tiedostojen synkronointiohjelma. Palvelimelle suunnitellaan myös verkon yli jaettua hakemistoa, johon käytetään SSHFS:ää. SSHFS on tiedostojärjestelmä, jonka avulla voidaan jakaa hakemisto verkon yli SSH-protokollaa käyttäen. Lopuksi palvelimelle tehdään rajoitettu varmuuskopiointi, joka olisi minimissään paikallisesti toiselle levylle tai tietokoneelle.

3 LINUX-PALVELIMEN ASENNUS YLEISESTI

Tässä luvussa käsittelen Linux-palvelimen asennusta yleisestä näkökulmasta. Pyrin antamaan lukijalle kokonaisvaltaisen kuvan siitä, miten Linux toimii ja mitä seikkoja tulee ottaa huomioon sen konfiguroinnissa palvelinkäyttöä varten. Oletan, että lukija ei tiedä paljoa Linuxin toiminnasta, ja pyrin selittämään suoritetut komennot ja konfiguraatioparametrit mahdollisimman selvästi.

3.1 Linux-jakelut ja sen valitseminen

Linux-käyttöjärjestelmän asentaminen alkaa sopivan jakelun valinnalla. Linuxin avoimuuden takia siitä on luotu monia eri versioita, joita nimitetään jakeluiksi (distribution). Jakeluilla on yhteistä se, että ne käyttävät samaa Linux-ydintä, mutta muuten ne ovat omia, uniikkeja kokonaisuuksiaan, joilla on omat erikoistumisalueensa. Ehdottomasti suosituimpia Linux-jakeluita ovat Debian ja siihen pohjautuvat jakelut. W3Techin päivittäisen seurannan mukaan suosituimpia Linux-jakeluita heidän testaamistaan palvelimista ovat Debian-pohjainen Ubuntu 34,9 prosentilla, alkuperäinen Debian 15,5 prosentilla sekä yrityskäyttöön tarkoitettu CentOS 9,9 prosentilla. (W3Techs 2021)

Debianin ja Ubuntun suosioon vaikuttavat niiden luotettavuus, turvallisuus sekä pitkäaikainen tuki. (Debian 2021) Erityisesti palvelinkäytössä nämä seikat korostuvat, sillä palvelimet ovat usein päällä päiviä tai jopa kuukausia yhtäaikaisesti, jolloin on tärkeää, että järjestelmä on mahdollisimman stabiili ja että päivitykset ovat saatavilla nopeasti. Jakelun valintaan vaikuttaa myös tarvittavat työkalut. Esimerkiksi Ajapaik käyttää palvelimellaan Ubuntua myös sen takia, että se on NVIDIA:n työkalujen pääjakelu. Tässä opinnäytetyössä tulenkin perehtymään Ubuntu-käyttöjärjestelmään, ja suosittelen sitä myös Linux-alkajille. Antamani ohjeet kuitenkin pätevät suurimpaan osaan muita Linux-jakeluita. En tule perehtymään opinnäytetyössä tarkemmin jakelun asennusprosessiin, sillä se voi vaihdella jakeluiden välillä ja jakeluiden kotisivuilla on myös yleensä laajat ohjeistukset, miten asennus tehdään.

3.2 Linux-komentojonon käyttö

Komentojonoon tulee törmäämään pakosti, mikäli tekee töitä Linux-pohjaisten palvelimien kanssa. Yleensä palvelimet pyörivät niin kutsutussa ”headless”-tilassa, jossa palvelimella ei pyöri minkäänlaista graafista käyttöliittymää, ja hallinta on tehtävä tekstipohjaista komentojonoa käyttäen. Komentojono voi olla aloittelijalle hyvin monimutkaisen ja hämmentävän tuntuinen, mutta harjoittelun ja totuttelemisen jälkeen se on jopa nopeampi tapa tehdä asioita kuin graafinen käyttöliittymä.

Komentojonoa kutsutaan Linux-jakeluissa yleensä terminaaliksi tai terminaaliemulaattoriksi. Terminaalin avatessa käyttöjärjestelmä käynnistää komentotulkin (shell). Komentotulkki on ohjelma, joka mahdollistaa komentojen suorituksen komentojonon kautta. Komennot voivat joko olla toisia ohjelmia tai komentotulkin omia funktioita. (Ward 2021, 12) Komentotulkkia voidaan myös käyttää skriptien tekoon. Oletuksena komentotulkki on suurimmassa osassa Linux-jakeluista bash-tulkki, jota tulen myös käyttämään tässä opinnäytetyössä.

Bash-komentotulkki näyttää oletuksena vasemmalta oikealle käyttäjänimen, tietokoneen nimen, työhakemiston sekä käyttäjäoikeudet, jotka merkataan joko dollarikuviolla tai risuaidalla. Dollarikuvio tarkoittaa, että käyttäjäoikeudet ovat rajatut, ja risuaita tarkoittaa, että ne ovat rajattomat. Kerron tarkemmin käyttäjäoikeuksista myöhemmässä kappaleessa. Kun bash käynnistetään, se on useimmiten oletuksena aaltoviivahakemistossa, joka on Linuxissa käyttäjän oman kotihakemiston symboli. (Kuvio 1)

```
siru@Ubuntu:~$
```

Kuvio 1. Esimerkki bash-tulkin oletuskehotteesta.

Ohjelmia suoritetaan tulkissa yksinkertaisesti kirjoittamalla sen nimi, mahdolliset liput ja parametrit sekä lopuksi painamalla enter-näppäintä. Liput tulevat yleensä ensimmäisenä ohjelman nimen jälkeen, ja ne yleensä sisältävät viivan ja kirjaimen. Lippujen avulla voidaan antaa ohjelmalle lisätietoa ja muuttaa sen käytöstä. Yleensä liput eivät ole pakollisia, vaan niillä voidaan esimerkiksi saada ohjelma antamaan enemmän lokitietoa ongelmatilanteiden ratkomiseksi. Parametrit kirjoitetaan yleensä lippujen jälkeen, ja ne antavat ohjelmalle tiedon siitä, mitä sen kuuluu tehdä. Parametrit ovat pakollisia esimerkiksi tiedostojen poistamisessa, jolloin parametriksi kuuluu antaa poistettavan tiedoston nimi.

3.3 Tiedostohierarkia Linuxissa

Linuxin tiedostohierarkia eroaa Windowsin hierarkiasta pääasiassa siten, että Windowsin erinäisten asemakirjainten kuten C ja D sijaan Linuxissa on vain yksi juurihakemisto, jonka sisältä kaikki tiedostot ja asemat löytyvät. Tätä juurihakemistoa merkitään kauttaviivalla. Kauttaviiva on myös Linuxissa hakemistojen erottajamerkki Windowsin kenoviivan sijasta. Juurihakemisto olisi siis /, alahakemisto juurihakemistossa /hakemisto ja tiedosto tämän hakemiston sisällä /hakemisto/tiedosto. Linux myös eroaa Windowsissa siinä, että Linuxissa jokainen asia on pääasiassa tiedosto juurihakemistossa. Hyvänä esimerkkinä voidaan ottaa /dev -hakemisto, joka sisältää tietokoneen laitetiedostot. Hakemistosta löytyy muun muassa /dev/random-laitetiedosto, joka luettuna tulostaa satunnaista dataa. Hakemistosta löytyy myös kaikki tietokoneeseen kytketyt laitteet, kuten kovalevyt ja CD-asemat. Linuxin mukana tulee tiettyjä oletushakemistoja, joiden tarkoitus voi olla aloittelijalle hämärä. Oheisessa taulukossa pyrin selittämään nämä hakemistot ja niiden tarkoitukset. (Linux manual, 2022)

Taulukko 1. Linuxin oletushakemistot.

Hakemiston nimi	Hakemiston tarkoitus
/bin	Sisältää suoritettavia ohjelmia, joita tarvitaan järjestelmän peruskäyttöön, kuten cp, rm ja mv.
/boot	Sisältää tiedostoja, joita tarvitaan Linuxin käynnistymiseen.
/dev	Sisältää laitetiedostoja.
/etc	Sisältää ohjelmien konfiguraatiotiedostoja.
/home	Sisältää käyttäjien kotihakemistoja.
/lib	Sisältää ohjelmakirjastoja, joita tarvitaan Linuxin käynnistymiseen.
/mnt	Sisältää järjestelmään liitettyjä laitteita ja tiedostojärjestelmiä.
/opt	Sisältää lisäosapaketteja. Tänne voidaan asentaa esimerkiksi sellaiset ohjelmistot, joita ei saada asennettua Linux-jakelun omalla paketinhallintaohjelmistolla.
/proc	Sisältää tietoa mm. järjestelmän suorittavista prosesseista sekä muuta Linux-ytimen antamaa tietoa.
/root	Pääkäyttäjän kotihakemisto.
/sbin	Sisältää suoritettavia ohjelmia, joita tarvitaan järjestelmän käynnistymiseen.
/srv	Sisältää palvelimien omaa dataa, kuten esimerkiksi http-palvelimen html-tiedostoja.
/sys	Sisältää samaa, Linux-ytimen antamaa dataa kuin /proc-kansiossa, mutta paremmin järjesteltynä.
/tmp	Sisältää väliaikaisia tiedostoja
/usr/bin	Sisältää useimmat suoritettavat ohjelmat, joita ei tarvita järjestelmän peruskäyttöön.
/usr/share	Sisältää ohjelmien tarvitsevia tiedostoja, joita voidaan jakaa eri Linux-jakeluiden välillä.
/var	Sisältää tiedostoja, jotka voivat muuttaa kokoaan, kuten esimerkiksi lokitiedostot.

3.4 Navigointi ja tiedosto-operaatiot

Linux-komentojonossa nykyisen työhakemiston tiedostojen ja hakemistojen listaaminen onnistuu ls-komennolla. ls-komennon suorittaminen ilman parametrejä yksinkertaisesti listaa kaikki nykyisen työhakemiston alahakemistot ja tiedostot lukuun ottamatta piilotettuja tiedostoja. Yleensä ne ovat myös värikoodattuja; ei-värjätyt ovat normaaleja tiedostoja, vihreät ovat suoritettavia tiedostoja ja siniset ovat hakemistoja. (Kuvio 2)

```
siru@Ubuntu:~$ ls
6.JPG  FSRCNN_x2.pb  Kuva.png
```

Kuvio 2. ls-komennon esimerkki.

Jos halutaan nähdä myös piilotetut tiedostot, voidaan ls-komennolle antaa parametriksi -a. Tällöin nähdään myös kaikki niin kutsutut pistetiedostot ja -hakemistot. Linuxissa tiedostosta tulee piilotettu, kun sen edessä on piste. (Kuvio 3)

```
siru@Ubuntu:~$ ls -a
.              .bash_logout  .config    .motd_shown     .sudo_as_admin_successful  6.JPG
..             .bashrc       .landscape .profile        .viminfo                   FSRCNN_x2.pb
.bash_history  .cache        .local     .python_history .wget-hsts                 Kuva.png
```

Kuvio 3. ls-komennon esimerkki 2.

Jokaisessa hakemistossa on myös kaksi erikoishakemistoa, piste- ja tuplapiste-hakemistot. Nämä ovat symboleja aaltoviivan tapaan. Piste-hakemisto on symboli tämänhetkiselle hakemistolle, ja tuplapiste-hakemisto taas yhtä hakemistoa

ylemmälle hakemistolle tiedostohierarkiassa. Työhakemiston vaihtaminen onnistuu cd-komennolla, jolle annetaan parametriksi halutun hakemiston nimi. Tämä voi olla absoluuttinen nimi, jolloin parametri on hakemiston koko nimi tiedostohierarkiassa, tai relatiivinen nimi, jolloin parametri on hakemisto nykyisessä työhakemistossa. Esimerkiksi *cd kansio*-komento suoritettuna /home/siru-hakemistossa siirtäisi työhakemiston hakemistoon /home/siru/kansio, kun taas *cd /bin*-komento siirtäisi työhakemiston hakemistoon /bin.

Tiedostojen kopiointiin, poistamiseen, siirtämiseen ja uudelleennimeämiseen on kaikkiin omat työkalunsa. Tiedostojen kopiointi onnistuu cp-ohjelmalla, jolle annetaan parametreiksi kopioitava tiedosto tai tiedostot sekä sijainti, johon ne kopioidaan. Kopioidulle tiedostolle voidaan myös antaa uusi nimi. cp ei oletuksena siirrä hakemistojen koko sisältöjä, vaan se pitää erikseen määritellä -r-lipulla. Esimerkiksi *cp -r /usr/share/tiedostot/ /home/siru* kopioisi hakemistossa /usr/share sijaitsevan hakemiston nimeltä tiedostot hakemistoon /home/siru. Tiedostoja voidaan poistaa rm-komennolla, jolle annetaan parametriksi poistettava tiedosto tai tiedostot. rm ei suostu oletuksena poistamaan hakemistoja cp:n tapaan, vaan se pitää myös määritellä -r-lipulla. Esimerkkinä komento *rm -r tiedosto1 tiedosto4 hakemisto/* poistaisi nykyisestä työhakemistosta tiedostot nimeltä tiedosto1 ja tiedosto4, sekä hakemiston nimeltä hakemisto ja kaikki tiedostot sen sisällä. Tiedostojen siirtämiseen ja uudelleennimeämiseen voidaan käyttää mv-komentoa, jonka parametriksi annetaan siirrettävän tiedoston nimi sekä sen uusi sijainti. Siirrettävälle tiedostolle voidaan myös määritellä uusi nimi. Esimerkiksi komento *mv Pictures/ Kuvat/* uudelleennimeäisi nykyisessä työhakemistossa sijaitsevan hakemiston Pictures sen suomenkieliseen vastineeseen.

3.5 Käyttäjät ja käyttöoikeudet

Linux-järjestelmässä käyttäjä määritellään entiteettinä, joka pystyy suorittamaan prosesseja ja omistamaan tiedostoja. Käyttäjiin viitataan käyttäjänimellä. Jokaisella käyttäjällä on myös oma, uniikki numerotunnuksensa, jota kutsutaan UID:ksi (user ID). Jokaisella käyttäjällä on omat oikeutensa ja rajansa. Esimerkiksi jokainen

käyttäjä pystyy muokkaamaan vain omia tiedostojaan ja prosessejaan. Käyttäjien lisäksi Linuxissa on olemassa myös ryhmät. Käyttäjät voivat kuulua ryhmiin, jotka pystyvät jakamaan oikeuksia keskenään. (Ward 2021, 9) Jokaisella ryhmällä on myös oma tunnuksensa, jota kutsutaan GID:ksi (group ID). Jokaisella käyttäjällä on yleensä myös oma ryhmänsä, jolla on sama nimi kuin käyttäjänimellä. Oman käyttäjän UID ja ryhmät voidaan nähdä suorittamalla id-komento.

3.5.1 Pääkäyttäjä ja oikeuksien korotus

Linuxissa järjestelmänvalvojan tunnusta kutsutaan pääkäyttäjäksi. Pääkäyttäjän nimi on yleensä root, ja sillä on oikeudet nähdä ja muokata kaikkia tiedostoja ja prosesseja järjestelmässä. Tämän takia root-käyttäjänä tulisi olla erityisen varovainen, sillä sitä käyttämällä on mahdollista aiheuttaa järjestelmään pysyvää vahinkoa. Tavallisen käyttäjän on mahdollista suorittaa komentoja pääkäyttäjänä käyttämällä sudo-ohjelmaa. sudo vaatii toimiakseen sen, että haluttava käyttäjä on lisätty sudo:n sallittujen käyttäjien listaan. Tämä yleensä tapahtuu käyttöjärjestelmän asennuksen yhteydessä käyttäjätunnuksen luonnissa. Tämän jälkeen sudo:a voidaan käyttää yksinkertaisesti laittamalla se sen komennon eteen, joka halutaan suorittaa pääkäyttäjän oikeuksilla. sudo kysyy suorittavan käyttäjän salasanan, ja suorittaa komennon. Mikäli root-käyttäjällä halutaan suorittaa enemmän komentoja, voidaan sille kirjautua su-ohjelmalla. su kysyy pääkäyttäjän salasanan, jonka jälkeen se kirjautuu tunnukselle ja käynnistää oletuskomentotulkin. root-käyttäjältä voidaan tämän jälkeen kirjautua ulos suorittamalla exit-komento.

3.5.2 Tiedosto-oikeudet

Jokaisella tiedostolla ja hakemistolla on Linuxissa omat oikeutensa. Nämä oikeudet voidaan nähdä antamalla ls-komennolle lippu *-l*. Tämä listaa nykyisen hakemiston tiedostot näyttäen myös niiden oikeudet, omistavan käyttäjän ja ryhmän, koon tavuina sekä viimeisimmän muokkausajan. (Kuvio 4)

```
siru@Ubuntu:/etc$ ls -l
total 357
drwxr-xr-x 1 root root      512 Feb 20  2021 NetworkManager
drwxr-xr-x 1 root root      512 Feb 20  2021 PackageKit
drwxr-xr-x 1 root root      512 Feb 20  2021 X11
-rw-r--r-- 1 root root     3028 Feb 20  2021 adduser.conf
```

Kuvio 4. ls -l-komennon esimerkki

Oikeudet määritellään yhdeksänä bittinä. Ensimmäiset kolme viittaavat omistavan käyttäjän oikeuksiin, seuraavat kolme omistavan ryhmän oikeuksiin, ja viimeiset kolme kaikkien muiden oikeuksiin. Näistä kolmesta bitistä ensimmäinen viittaa lukuoikeuteen, toinen kirjoitusoikeuteen ja kolmas suoritusoikeuteen. ls-komennossa on myös oikeuksien edessä yksi ylimääräinen kirjain, joka tarkoittaa tiedoston tyyppiä. Tämä on viiva normaalin tiedoston tapauksessa, mutta esimerkiksi hakemistoissa se on d-kirjain. (Negus 2020, 105–106)

Oheisessa esimerkissä nähdään, että ensimmäinen tiedosto /etc-hakemistossa on NetworkManager. Ensimmäisestä kirjaimesta selviää, että kyseessä on hakemisto. Tämän jälkeisistä omistavan käyttäjän oikeuksista nähdään, että tiedoston omistajalla on oikeudet lukea, kirjoittaa ja suorittaa. Omistavan ryhmän jäsenillä ja kaikilla muilla taas on oikeudet vain lukea ja suorittaa. Hakemiston omistava käyttäjä on root, ja omistava ryhmä root. adduser.conf-tiedoston tapauksessa nähdään,

että se on normaali tiedosto, jonka omistajalla on oikeudet lukea ja kirjoittaa tiedostoa, kun taas omistavalla ryhmällä ja kaikilla muilla on oikeudet vain lukea. (Kuvio 4)

Tiedostojen oikeuksia voidaan muokata chmod-komennolla. Oikeuksia muokatakseen käyttäjän tulee olla joko tiedoston omistaja tai pääkäyttäjä. chmod ottaa parametreiksi halutut oikeudet sekä muokattavan tiedoston nimen. Oikeudet annetaan kolmena numerona, joista ensimmäinen vastaa uusia omistavan käyttäjän oikeuksia, toinen omistavan ryhmän oikeuksia sekä kolmas kaikkien muiden oikeuksia. Nämä numerot voivat olla nollan ja seitsemän väliltä, joista nolla tarkoittaa ei oikeuksia, ja 7 tarkoittaa luku- kirjoitus- ja suoritusoikeuksia. Oheisessa taulukossa listaan nämä numerot, sekä sen, mitä oikeuksia ne antavat. (Linux.fi, 2022)

Taulukko 2. chmod-komennon mahdolliset oikeudet.

Numero	Näkymä ls-komennossa	Oikeudet
0	---	Ei oikeuksia
1	--x	Suoritus
2	-w-	Kirjoitus
3	-wx	Kirjoitus ja suoritus
4	r--	Luku
5	r-x	Luku ja suoritus
6	rw-	Luku ja kirjoitus
7	rwx	Luku, kirjoitus ja suoritus

Esimerkiksi komento *chmod 754 tiedosto* antaisi tiedoston omistajalle luku-, kirjoitus- ja suoritusoikeudet, omistavalle ryhmälle luku- ja suoritusoikeudet, ja kaikille muille vain lukuoikeudet. Tiedoston omistajia voidaan taas muuttaa chown-

komennolla, jonka käyttö on yksinkertaisempaa. chown-komennolle annetaan parametriksi uusi omistajakäyttäjä ja omistajaryhmä erotettuina kaksoispisteellä, sekä muutettavan tiedoston nimi. Esimerkiksi komento *chown siru:koulu tiedosto* muuttaisi tiedoston omistajaksi käyttäjän siru ja omistajaryhmäksi ryhmän koulu.

3.6 Paketinhallinta ja ohjelmistojen asennus

Paketinhallinta on pääasiallinen tapa asentaa uusia ohjelmia Linux-käyttöjärjestelmään. Paketinhallintaohjelmalla pystytään asentamaan, poistamaan ja päivittämään paketteja, jotka sisältävät ohjelmia tai muita ominaisuuksia. Paketit ladataan yleensä jakeluiden omista pakettivarastoista (repository). Linux-jakeluissa on useita erilaisia paketinhallintaohjelmistoja, mutta Ubuntu sekä muut Debian-pohjaiset jakelut käyttävät apt-ohjelmaa, johon tulen perehtymään tässä opinnäytetyössä.

Apt (Advanced Packaging Tool) on Ubuntun käyttämä paketinhallintaohjelmisto, jonka avulla voidaan asentaa uusia ohjelmistopaketteja sekä päivittää ja poistaa nykyisiä. (Ubuntu 2021) Ennen uusien pakettien asentamista on suositeltavaa päivittää apt:n pakettitietokanta uusimpaan versioon. Tämä varmistaa, että asennetut paketit ovat uusimpia versioita. Pakettitietokanta päivitetään suorittamalla *apt* pääkäyttäjän oikeuksilla parametrilla *update*. Tämä lataa uusimmat pakettitietokannat jakelun pakettivarastoista, vertaa niitä nykyisiin asennettuihin paketteihin ja myös ilmoittaa, mikäli päivityksiä on saatavilla.

Uuden ohjelman asennus tehdään yksinkertaisesti suorittamalla *apt* pääkäyttäjän oikeuksilla parametreilla *install* sekä asennettavan ohjelman nimellä. Jos haluttaisiin esimerkiksi asentaa Firefox-selain, se onnistuisi komennolla *sudo apt install firefox*. Tämän komennon aikana *apt* ensin selvittää *firefox*-paketin mahdolliset riippuvuudet (dependencies). Riippuvuudet ovat ylimääräisiä paketteja, joita jokin toinen paketti tarvitsee toimiakseen. (Ubuntu 2021) Esimerkkinä *firefox*-paketti tarvitsee toimiakseen *libjpeg8*-paketin JPEG-kuvien näyttämiseen. Riippuvuudet

selvitettyään *apt* listaa asennettavat paketit ja sen, kuinka paljon levytilaa ne tulevat viemään. Käyttäjä voi vielä vahvistaa tai peruuttaa asennuksen tässä vaiheessa. Mikäli asennus vahvistetaan, *apt* lataa kaikki listatut paketit sekä asentaa ja mahdollisesti konfiguroi ne. Asennuksen jälkeen ohjelma voidaan suorittaa normaalisti komentotulkista tai ohjelmavalikosta, mikäli graafinen käyttöliittymä on käytössä.

Ohjelma voidaan taas poistaa *remove*-parametrilla pääkäyttäjän oikeuksilla samaan tapaan kuin ohjelman asennus toimi. *remove*-parametrissa tulee kuitenkin huomioida, että se ei poista kyseisen paketin mukana asennettuja riippuvuuksia tai konfiguraatiotiedostoja. Konfiguraatiot voidaan poistaa lisäämällä *remove*-parametrin perään lippu *--purge*. Apt pystyy myös poistamaan käyttämättömät, riippuvuuksina asennetut paketit parametrilla *autoremove*, mikä onkin hyvä suorittaa säännöllisesti levytilan vapauttamiseksi.

Järjestelmä voidaan päivittää parametrilla *upgrade*. Ennen päivittämistä on tärkeää muistaa ensin päivittää pakettitietokannat uusimpiin versioihin käyttämällä *update*-parametria. *upgrade* vertaa pakettitietokannassa olevien pakettien versioita nykyisten asennettujen pakettien versioihin, ja mikäli uusia versioita on tarjolla, ehdottaa päivitystä. Mikäli päivitys vahvistetaan, apt lataa uudet versiot ja asentaa ne, samalla ilmoittaen, jos jokin asia ohjelmistossa muuttuu tai tarvitsee ylimääräistä konfigurointia.

Mikäli asennettavan ohjelman tarkkaa pakettinimeä ei tiedetä, voi apt myös hakea paketteja *search*-parametrilla. *apt search firefox*-komennolla voidaan hakea kaikki ne paketit, jotka liittyvät jotenkin Firefoxiin, sekä lyhyt kuvaus siitä mitä ne ovat. Jos paketista tarvitaan tarkempaa tietoa, voidaan se hakea *show*-parametrilla. *show* näyttää muun muassa paketin riippuvuudet sekä tarkemman kuvauksen.

Vastaan voi tulla tilanne, jossa haettua ohjelmistoa ei löydy jakelun pakettivarastoista. Tällöin ohjelma on asennettava perinteisesti lataamalla se Internetistä. Tässä Debianin suosio on hyödyksi, sillä useimmat ohjelmistot ovat saatavilla .deb-

tiedostoina, jotka voidaan asentaa helposti dpkg-ohjelmalla. Asennus onnistuu suorittamalla *dpkg* lipulla *-i* ja parametrilla *ohjelmistopaketti.deb*. Vaihtoehtona .deb-tiedostoille on kolmannen osapuolen säiliön lisääminen järjestelmään. Tällöin ohjelmistot pystytään asentamaan ja päivittämään normaalisti apt:llä.

3.7 Prosessienhallinta

Linux on niin kutsuttu moniajojärjestelmä. Tämä tarkoittaa sitä, että se pystyy suorittamaan montaa ohjelmaa samaan aikaan. Näitä suorittavia ohjelmia kutsutaan prosesseiksi. (Negus 2020, 131) Jokaisella prosessilla on oma, uniikki numerotunnuksensa, joilla niihin voidaan viitata. Tätä tunnusta kutsutaan PID:iksi (Process ID). Linux-ydin tallentaa dataa prosesseista /proc-hakemistoon, josta ne löytyvät lajiteltuina alihakemistoihin prosessitunnuksen mukaan. Prosessinhallintatyökalut lukevat tätä dataa ja näyttävät sen perusteella listan prosesseista, niiden nimistä sekä niiden tiloista.

Yksinkertaisin työkalu prosessien näyttämiseen on ps, joka löytyy useimmista Linux-jakeluista esiasennettuna. Ilman annettuja parametreja ps listaa nykyisen käyttäjän käynnistämät prosessit, niiden prosessitunnuksen, prosessin käyttämän ajan sekä suorittavan ohjelman nimen. Jos prosesseista halutaan tarkempaa tietoa, voidaan ps:älle antaa parametriksi *u*. Tällöin ps listaa myös muun muassa prosessin omistajan, prosessin suoritinkäytön sekä prosessin muistin- ja virtuaalimuistinkäytön. Jos halutaan nähdä muutkin kuin käyttäjän omat prosessit, voidaan parametreihin lisätä *ax*. (Kuvio 5)

```
siru@Ubuntu:~$ ps axu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   8944   332 ?        Ssl  03:15   0:00 /init
root        12  0.0  0.0   8948   232 tty1     Ss   03:15   0:00 /init
siru        13  0.0  0.0  18076  3604 tty1     S    03:15   0:00 -bash
siru       198  0.0  0.0  18664  1880 tty1     R    03:33   0:00 ps axu
```

Kuvio 5. ps-komennon esimerkki

Prosesseja voidaan kontrolloida lähettämällä niille signaaleja. Näitä pystytään lähettämään esimerkiksi kill-ohjelmalla, joka nimestään poiketen pystyy lähettämään myös muitakin signaaleja kuin vain tapposignaaleja. Oletuksena kill lähettää prosessille TERM-signaalin, joka pyytää prosessia lopettamaan suorituksen antaen sille kuitenkin mahdollisuuden siivota jälkensä. kill toimii yksinkertaisesti antamalla sille parametrina halutun prosessin PID. Jos prosessi ei suostu sammumaan TERM-signaalilla, voidaan sille lähettää KILL-signaali. KILL-signaali pakottaa prosessin sammumaan antamatta sille mahdollisuutta siivota jälkiä tai tallentaa tiedostoja. Tämän takia sitä ei kannata käyttää muissa kuin pakkotilanteissa tietojen menetyksen välttämiseksi. KILL-signaali voidaan lähettää antamalla kill:ille lipuksi *-KILL*. Esimerkiksi *kill -KILL 45*-komento lähettäisi prosessille, jonka PID on 45 KILL-signaalin.

kill pystyy lähettämään myös muita signaaleja, jotka voivat olla hyödyksi erinäisissä tilanteissa. Koko listan signaaleista saa näytettyä antamalla kill:ille lipun *-l*. Yksi näistä signaaleista on STOP-signaali, jonka avulla voidaan jäädyttää prosessi nykyiseen tilaan. Tällöin se pysyy muistissa, mutta ei pysty tekemään mitään. Prosessin saa toimimaan jälleen lähettämällä sille CONT-signaalin. (Ward 2021, 33) Esimerkiksi komento *kill -STOP 45* jäädyttäisi prosessin, jonka PID on 45, ja komento *kill -CONT 45* saisi sen jatkamaan suoritusta.

Nämä ovat alkukantaisimpia tapoja hallita prosesseja, mutta Linux-jakeluihin saa asennettua myös toisia prosessinhallintaohjelmistoja, joiden avulla prosessien hallitseminen voi olla helpompaa. Yksi näistä on htop, jonka saa yleensä asennettua suoraan jakelun paketinhallintatyökalulla. htop näyttää yksinkertaisen, tekstipohjaisen käyttöliittymän, jossa näkyy tämänhetkinen prosessorin- ja muistinkäyttö, sekä lista prosesseista. Prosesseille saa lähetettyä signaaleja F9-näppäimellä, ja htop sulkeutuu F10-näppäimellä.

3.8 Init-järjestelmät ja järjestelmän hallinta

Init-järjestelmä on ohjelma, jonka päätarkoitus on käynnistää ja sammuttaa tarpeelliset prosessit ja palvelut järjestelmän käynnistyessä tai sammuessa. Tällä hetkellä kaikki isoimmat Linux-jakelut käyttävät init-järjestelmänä systemd-nimistä ohjelmistoa, jonka käyttöön tulen keskittymään. systemd on yksi uusimmista init-järjestelmistä, ja se pystyy perinteisistä init-järjestelmistä poiketen tarkkailemaan yksittäisiä prosesseja niiden käynnistymisen jälkeen, sekä ryhmittämään niitä keskenään. systemd pystyy myös prosessien lisäksi hallinnoimaan tiedostojärjestelmän liittämisiä (mount). (Ward 2021, 138–139)

systemd toimii niin kutsuttujen yksiköiden (unit) avulla. Nämä yksiköt määritellään unit-tiedostojen avulla, joissa kerrotaan haluttu tehtävä, kuten prosessin käynnistäminen ja monitorointi. Yksiköt voivat olla myös riippuvaisia toisistaan, jolloin jostain tietystä yksiköstä riippuvaiset prosessit eivät käynnisty ennen sitä yksikköä, josta ne ovat riippuvaisia. (Ward 2021, 138) Näin esimerkiksi Internet-yhteyttä vaativat palvelimet eivät yritä käynnistyä ennen verkkoyhteyden muodostamista. systemd tukee monia erilaisia yksikkömuotoja, mutta yleisimpiä niistä ovat palveluyksiköt (service unit), kohdeyksiköt (target unit), pistokeyksiköt (socket unit) sekä liitosyksiköt (mount units). (Ward 2021, 139) Näistä palveluyksiköt ovat kaikista yleisimpiä, ja tulenkin suurimmalta osin keskittymään niihin.

systemd:lle voidaan lähettää komentoja systemctl-komennolla. Tällä komennolla voidaan esimerkiksi aktivoida palveluita, tarkistaa niiden tila ja ladata konfiguraatiot uudelleen. Näin esimerkiksi juuri luodut yksiköt saadaan ladattua systemd:n muistiin. Komento *systemctl list-units* listaa kaikki järjestelmässä olevat yksiköt, niiden tyypin sekä niiden tilan ja kuvauksen. Jonkun tietyn yksikön tarkempi tila voidaan tarkistaa komennolla *systemctl status <yksikön nimi>*. Tämä listaa yksikön nimen, kuvauksen, sekä sen, onko yksikkö aktiivinen ja mikäli on, sen aktiiviset prosessit ja niiden PID:t. Yksikkö voidaan sammuttaa komennolla *systemctl stop <yksikön nimi>*, ja käynnistää komennolla *systemctl start <yksikön nimi>*. Mikäli yksikön halutaan käynnistyvän järjestelmän käynnistyessä, voidaan se aktivoida komennolla *systemctl enable <yksikön nimi>*, ja vastaavasti deaktivoida komennolla *systemctl disable <yksikön nimi>*. Tämä on yksinkertaisin tapa hallinnoida järjestelmää systemd:llä, ja yleensä se on riittävää.

systemd:hen voidaan myös luoda omia yksiköitä tekemällä .service-tiedostoja /etc/systemd/system-hakemistoon. Tämä on hyödyllistä, mikäli ohjelmiston mukana ei tule valmista systemd-yksikkötiedostoa. En mene tässä opinnäytetyössä tarkemmin unit-tiedostojen luontiin, mutta siihen löytyy hyviä ohjeita Internetistä. Kun uusi .service-tiedosto on tehty, se voidaan ladata systemd:n muistiin komennolla *systemctl daemon-reload*. Tällöin uutta yksikköä voidaan hallinnoida normaalisti yllä olevilla komennoilla.

3.9 Etäkäyttö

Etäkäyttö on erityisen tärkeää palvelinkäytössä. Sen avulla palvelinta voidaan hallinnoida ilman, että siihen on fyysistä pääsyä. Hallinnointi tehdään Linuxissa yleensä SSH-protokollaa käyttäen. SSH on turvallinen etähallintaprotokolla, joka käyttää julkisen avaimen salausta. Jokaisella SSH-palvelimella on oma, uniikki julkinen avaimensa, joka lähetetään käyttäjälle, kun palvelimeen yhdistetään. Näin käyttäjä voi varmistaa, että on yhdistämässä oikealle palvelimelle. SSH tukee myös turvallista tiedostojensiirtoa SFTP-protokollalla. (Secure File Transfer Protocol) SSH-palvelin voidaan asentaa Ubuntuun apt:lla. Komento *apt install ssh* asentaa

sekä SSH-palvelimen että SSH-asiakasohjelman. apt asentaa SSH-palvelimen systemd-yksikön, jonka avulla palvelin voidaan käynnistää ja sammuttaa systemd:tä käyttäen. Oletuksena asennuksen yhteydessä apt käynnistää SSH-palvelimen ja aktivoi systemd-yksikön, jolloin SSH-palvelin käynnistyy järjestelmän käynnistyessä automaattisesti. Oletuksena SSH-palvelin käynnistyy portissa 22, ja sallii kaikki kirjautumiset.

SSH-palvelimen käytössä täytyy kuitenkin ottaa tietoturvaseikkoja huomioon, mikäli palvelimeen pystytään yhdistämään julkisesta verkosta eli WAN:ista. Koska SSH-palvelin sallii kaikki kirjautumiset, voidaan sillä myös koittaa kirjautua pääkäyttäjänä sisään. Tämä on iso tietoturvariski, sillä jos pääkäyttäjälle murtaudutaan, voidaan järjestelmästä varastaa tietoja tai sinne voidaan asentaa haittaohjelmia. Tämän takia onkin suotavaa muokata SSH-palvelin konfiguraatiota niin, ettei pääkäyttäjällä kirjautuminen ole mahdollista.

Toinen tietoturvan kannalta tärkeä asia, joka kannattaa muuttaa, on kirjautuminen salasanoilla. Koska SSH tukee julkisen avaimen salausta, on siihen mahdollista tunnistautua avaintiedostolla salasanan sijaan. Nämä avaintiedostot mahdollistavat hyvin vahvan suojauksen, joka on paljon vahvempi kuin jopa erittäin pitkät salasanat. Avaintiedostoissa on myös se etu, että käyttäjien ei tarvitse muistaa salasanoja, ja he voivat myöskin käyttää samaa avainta useammalla SSH-palvelimella kirjautumiseen. (SSH.COM 2022) Julkisen avaimen salaus toimii luomalla kaksi avainparia, jotka ovat nimeltään julkinen avain ja yksityinen avain. Julkinen avain säilytetään palvelimella, ja sitä voi jakaa turvallisesti eteenpäin. Yksityinen avain taas tulisi säilyä vain käyttäjällä itsellään, eikä sitä saisi jakaa kenellekään. Tunnistautuessa käyttäjä lähettää jaetun salaisuuden palvelimelle, jonka palvelin pystyy varmentamaan julkisen avaimen avulla. Mikäli varmennus onnistuu, antaa palvelin käyttäjän kirjautua sisään.

3.9.1 SSH-avaimien luonti

SSH-avainpareja voidaan luoda ssh-keygen-ohjelmalla, joka tulee Ubuntussa ssh-paketin mukana. ssh-keygenille ei tarvitse antaa parametreja, jolloin se ensin kysyy tallennettavan tiedoston nimeä. Oletuksena avain tallennetaan käyttäjän kotihakemistossa sijaitsevaan, piilotettuun .ssh-hakemistoon. Käyttäjä voi myös määritellä avaimelle salasanan, jolloin käyttäjän on annettava tämä salasana aina avainta käytettäessä. Tämän jälkeen ssh-keygen luo avainparin, jossa oletuksena yksityinen avain löytyy id_rsa-tiedostosta, ja julkinen id_rsa.pub-tiedostosta. ssh-keygen löytyy myös esiasennettuna uusimmista Windows-versioista, ja sitä voidaan käyttää samalla tavalla Windowsin komentojonolta tai PowerShellista. Mikäli ssh-keygen ei ole Windowsissa saatavilla, voidaan käyttää PuTTY-ohjelmaa, joka on avoimen lähdekoodin SSH-asiakasohjelma Windowsille. PuTTY:ssä tulee mukana PuTTYgen-ohjelma, jolla avainpareja voidaan luoda.

Kun avainparit on luotu, täytyy julkinen avain määrittää kirjautumista varten. Tämä tapahtuu laittamalla julkinen avain tiedostoon nimeltä authorized_keys, joka sijaitsee .ssh-hakemistossa sen käyttäjän kotihakemistossa, jolle halutaan kirjautua avaimella. Esimerkiksi siru-nimisen käyttäjän tapauksessa julkinen avain olisi tiedostossa */home/siru/.ssh/authorized_keys*. Jos siis edellinen, ssh-keygenillä luotu avain haluttaisiin määrittää kirjautumiseen, voitaisiin yksinkertaisesti id_rsa.pub-tiedosto nimetä uudelleen authorized_keys:iksi komennolla *mv id_rsa.pub authorized_keys*. authorized_keys-tiedoston oikeuksia tulee myös säätää niin, että vain tiedoston omistaja pystyy lukemaan ja kirjoittamaan sitä. Tämä onnistuisi komennolla *chmod 600 authorized_keys*.

3.9.2 SSH-palvelimelle kirjautuminen

SSH-palvelimelle voidaan kirjautua Linuxissa ja uusimmissa Windows-versioissa ssh-komennolla. Pakolliset parametrit, joita ssh-komennolle tulee antaa, on käyttäjän nimi, jolla kirjaudutaan sekä kirjauduttavan palvelimen osoite erotettuna @-merkillä. Hyödyllisiä lippuja ovat -p, joka määrittää yhdistettävän portin, ja -i, joka

määrittää käytettävän avaintiedoston. Esimerkiksi komento *ssh -p 5496 -i avain.key siru@palvelin.com* yhdistäisi siru-käyttäjänä palvelin.com-osoitteeseen porttiin 5496, ja käyttäisi tunnistamiseen yksityistä avainta avain.key-nimisestä tiedostosta. Mikäli ssh-komento ei ole saatavilla, tai halutaan käyttää graafista ohjelmaa, voidaan käyttää esimerkiksi edellä mainittua PuTTY-ohjelmaa. PuTTY:n konfiguraatioikkunaan laitetaan palvelimen osoite ja portti. Yksityinen avain voidaan määrittää menemällä Connection-asetusten kohtaan SSH, josta löytyy Auth-asetukset. Yksityinen avaintiedosto määritellään valitsemalla se kohtaan ”Private key file for authentication”. Kun palvelimelle yhdistetään ensimmäistä kertaa, ssh varoittaa, että palvelimen julkinen avain on tuntematon. Tämä on normaalia, ja palvelin voidaan tallentaa ssh:n muistiin, jolloin siitä ei enää varoiteta, ellei palvelimen julkinen avain muutu jostain syystä.

3.9.3 SSH-palvelimen konfigurointi

SSH-palvelimen konfiguraatiotiedosto sijaitsee Ubuntussa /etc/ssh/sshd_config-tiedostossa. Oletuksena asetusten edessä on risuaita, joka tarkoittaa, ettei SSH-palvelin huomioi niitä. Jos asetusta halutaan muuttaa, voidaan risuaita poistaa ja asetuksen arvo muuttaa. Asetuksia, joita suosittelen muuttamaan WAN-verkkoa varten, on PasswordAuthentication, joka tulisi laittaa arvoon no. Tällöin palvelimelle voidaan kirjautua vain avaintiedostoilla. Toinen on PermitRootLogin, joka tulisi laittaa arvoon no, jolloin pääkäyttäjälle ei voida kirjautua SSH:n kautta. Nämä ovat tärkeimmät asetukset, jotka vähentävät tietoturvariskiä huomattavasti. Lisätietoa kaikista mahdollisista asetuksista ja siitä, mitä ne tekevät löytyy esimerkiksi ssh.com-sivustolta, jossa ne on listattu hyvin. Kun asetukset on muutettu, tulee ssh-palvelin käynnistää uudelleen systemd:llä. Tämä onnistuu suorittamalla komento *systemctl restart sshd* pääkäyttäjän oikeuksilla.

3.10 Palomuuri

Palomuuri on ohjelma, jolla voidaan estää haitallisen tai epähalutun datan pääseminen järjestelmään tietoverkkojen kautta. Se on etenkin palvelinkäytössä tärkeää olla asennettuna ja konfiguroituna, sillä muutoin esimerkiksi julkisesta verkosta voidaan ottaa yhteyttä sellaisiin palvelimiin, joihin ei haluta pääsyä sisäverkon ulkopuolelta. Linuxissa palomuuri toimii ydintasolla iptables-ohjelman kautta. iptables toimii sille määritettyjen sääntöjen avulla, jotka voidaan määrittää koskemaan joko lähtevää tai saapuvaa liikennettä. iptablesiin on myös mahdollista määrittää uudelleenohjaussääntöjä, joilla esimerkiksi tietystä osoitteesta tuleva liikenne voidaan uudelleenohjata johonkin toiseen laitteeseen verkon sisällä. (Negus 2020, 672–673) iptablesin käyttö voi olla varsinkin aloittelijalle hämmentävää, ja siksi perehdynkin opinnäytetyössä ufw-ohjelmaan (Uncomplicated Firewall). ufw on yksinkertaistettu tapa tehdä iptables-sääntöjä, ja se löytyy esiasennettuna Ubuntusta. ufw:n tila voidaan tarkistaa suorittamalla komento *ufw status* pääkäyttäjän oikeuksilla. Oletuksena ufw on poissa päältä, ja komento sanoo ufw:n olevan epäaktiivinen. ufw voidaan käynnistää suorittamalla komento *ufw enable*. Tämän jälkeen palomuuri aktivoituu, sekä myöskin käynnistyy automaattisesti järjestelmän käynnistyessä. ufw voidaan vastaavasti sammuttaa komennolla *ufw disable*.

Oletuksena ufw sallii kaiken liikenteen, jolle ei ole erikseen määritelty sääntöä. Tätä voidaan muuttaa komennolla *ufw default <sääntö> <suunta>*. Mahdollisia sääntöjä on *allow*, joka sallii kaiken liikenteen, *deny*, joka estää kaiken liikenteen ja *reject*, joka estää liikenteen, samalla ilmoittaen yhdistävälle osapuolelle, että liikenne estettiin. Mahdollisia suuntia on *incoming*, joka tarkoittaa saapuvaa liikennettä, *outgoing*, joka tarkoittaa lähtevää liikennettä ja *routed*, joka tarkoittaa uudelleenohjattua liikennettä. Esimerkiksi komento *ufw default deny incoming* konfiguroisi palomuurin niin, että se oletuksena estää kaiken saapuvan liikenteen, ellei sille erikseen ole määritelty sääntöä. Tämä on suositeltavaa palvelinkäytössä, sillä se vähentää tietoturvariskejä. (Rothman 2010)

Jos jokin tietty portti halutaan sallia palomuurin läpi, sille voidaan lisätä salliva sääntö komennolla *ufw allow <porttinumero>*. Esimerkiksi komento *ufw allow 80* sallisi liikenteen portin numero 80 kautta, joka on standardi portti HTTP-yhteyksille. Liikenne voidaan taas kieltää deny-komennolla, joka toimii samalla tavalla kuin aiempi allow-komento. Kun palomuuri on aktivoitu ja siihen on määritelty sääntöjä, näyttää *ufw status*-komento kaikki palomuuriin määritellyt säännöt. Sääntöjä voidaan poistaa yksinkertaisesti lisäämällä delete-komento saman säännön eteen. Esimerkiksi komento *ufw delete allow 80* poistaisi edellisen esimerkin säännön. Mikäli liikenne halutaan sallia vain tietystä IP-osoitteesta, voidaan säännön perään lisätä *from <IP-osoite>*. Näin komento *ufw allow 80 from 192.168.1.50* sallisi liikenteen porttiin 80 vain osoitteesta 192.168.1.50. Vastaavasti liikenne voidaan sallia vain tietystä aliverkosta lisäämällä IP-osoitteen eteen verkkomaski. Esimerkiksi *ufw allow 80 192.168.1.0/24* sallisi liikenteen porttiin 80 vain IP-osoitteista, jotka ovat väliltä 192.168.1.0–192.168.1.255. (Ubuntu 2022)

ufw sisältää myös valmiita konfiguraatioita ohjelmistoille, joiden avulla on mahdollista nopeasti konfiguroida kaikki tietyn ohjelmiston tarvitsemat portit. Asennetut ohjelmistokonfiguraatiot voidaan listata komennolla *ufw app list*, ja niistä voidaan nähdä tarkempaa tietoa komennolla *ufw app info <nimi>*. Näitä ohjelmistojen nimiä voidaan käyttää aiemmissa ufw-komennoissa porttinumeroiden sijaan. Esimerkiksi komento *ufw allow OpenSSH* sallisi kaikki SSH-palvelimen tarvitsemat portit.

4 ANALYTIIKKAPALVELIMEN ASENNUS

Tässä kappaleessa kerron Ajapaikin analytiikkapalvelimen asennuksesta ja konfiguroinnista. Tarkempaa tietoa projektin taustoista kerroin aiemmin kappaleessa 2. Asennus aloitettiin asentamalla palvelimelle Ubuntu-jakelun versio 21.04. Tämä kuitenkin osoittautui hankalaksi, sillä tuoreimmat NVIDIA:n ajurit olivat tarjolla heidän pakettivarastoistaan vain Ubuntun versiolle 20.04. Tämä tarkoitti sitä, että ajurit oli asennettava manuaalisesti NVIDIA:n sivulta ladatulla paketilla. Tämä teki asioista hankalampia, sillä ajureiden päivittäminen ja mahdollinen poistaminen tulisi olemaan hieman vaikeampaa. Tällä versiolla kuitenkin jatkettiin, ja ensiksi aloitettiin asentamaan koneoppimisessa tarvittavaa OpenCV:tä. Palvelimelle asennettiin aivan aluksi SSH-palvelin, ja se konfiguroitiin aiemmin mainitsemillani esimerkeillä estämään pääkäyttäjälle kirjautuminen, ja pakottamaan avaintiedostojen käytön tunnistautumisessa. Ajapaikin pääpalvelimelle asennettiin myös niin kutsuttu SSH-tunneli, jonka avulla pystyttiin uudelleenohjaamaan tietylle portille saapuva liikenne Ajapaikin pääpalvelimelta analytiikkapalvelimelle.

4.1 CUDA & OpenCV:n kääntäminen

Kuten aiemmin luvussa 2 kerroin, OpenCV on avoimen lähdekoodin konenäkökirjasto. OpenCV on saatavilla Ubuntun virallisista pakettivarastoista, mutta niistä puuttuu tarvittavia lisäosamoduuleita, joita tarvitaan muun muassa kuvien ylösskaalaukseen koneoppimisen avulla. Tämän takia OpenCV piti kääntää lähdekoodeista. Ohjelmiston kääntämisessä se muutetaan luettavasta lähdekoodista binaarimuotoon, jota tietokoneen prosessori pystyy suorittamaan. Kääntäminen onnistui lataamalla OpenCV:n lähdekoodit sekä ekstramoduulien lähdekoodit omiin hakemistoihinsa. Hakemistoon luotiin myös yksi uusi build-niminen hakemisto.

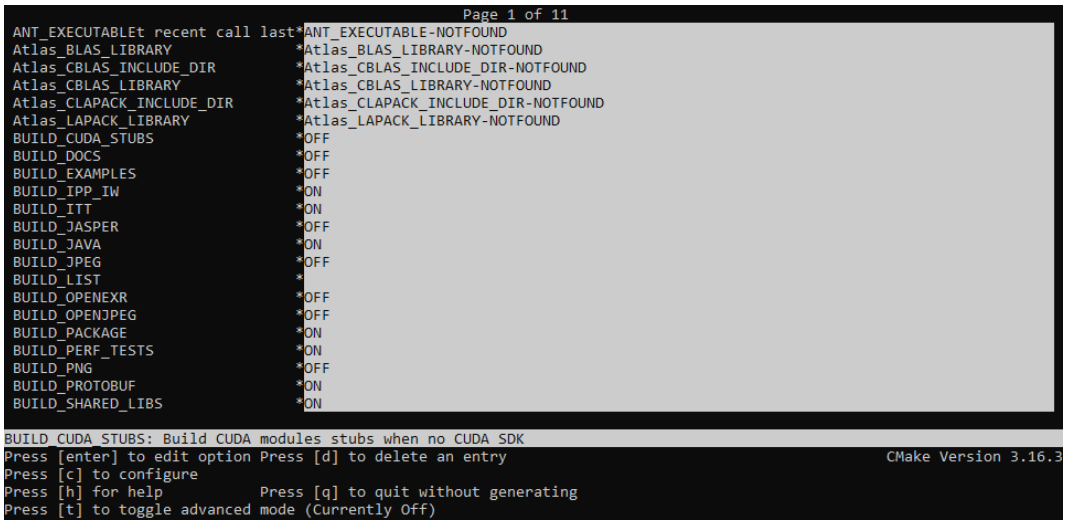
Tämän jälkeen lähdekoodit tulee konfiguroida, jolloin voidaan muokata, mitä ominaisuuksia ohjelmassa on. Konfigurointi myös varmistaa, että kaikki tarvittavat kirjastot ja ohjelmat mitä ohjelman kääntämiseen tarvitaan ovat asennettu ja että ne toimivat. Konfigurointi tehdään yleensä lähdekoodien mukana tulevalla configure-

nimisellä ohjelmistolla, mutta tämä vaihtelee lähdekoodeista riippuen. Konfiguroinnin jälkeen konfigurointiohjelmisto tallentaa määritettyyn hakemistoon kääntöohjeet, joita kääntäjäohjelma tarvitsee lähdekoodin kääntämiseen binaariksi. Ennen konfiguroinnin aloittamista tuli kuitenkin asentaa NVIDIA:n CUDA-kirjastot, jotka mahdollistavat NVIDIA:n näytönohjaimien hyödyntämisen. Ilman CUDA-tukea OpenCV pystyy käyttämään vain tietokoneen prosessoria, jolloin tehtävissä menee hyvin kauan. CUDA-kirjastot sai asennettua virallisista säilöistä nimellä *nvidia-cuda-toolkit*, jonka jälkeen ne olivat valmiita käytettäviksi. Tarvittiin myös Python-ohjelmointikieli, jolloin OpenCV:tä voitaisiin käyttää sen kautta. Tämä asennettiin samalla tavalla apt:n kautta.

OpenCV käyttää lähdekoodin konfigurointityökaluna CMake-nimistä ohjelmistoa, joka on avoimen lähdekoodin järjestelmä, joka hallitsee ohjelmiston kääntöprosessia käyttöjärjestelmä- ja kääntäjäriippumattomasti. (CMake 2022) CMakella pystytään siis luomaan samasta lähdekoodista monille eri kääntäjäohjelmistoille sopivia kääntöohjeita. Näin OpenCV voitaisiin esimerkiksi kääntää sekä Linux- että Windows-käyttöjärjestelmille samasta lähdekoodista. CMake saatiin asennettua Ubuntuun helposti apt:n avulla. Tämän jälkeen lähdekoodi oli valmis konfiguroitavaksi.

Konfigurointiin käytettiin CMaken mukana tulevaa ccmake-sovellusta, joka on tekstipohjainen versio CMaken graafisesta käyttöliittymästä. ccmake:lle annetaan parametriksi hakemisto, jossa lähdekoodit ovat. ccmake:lle annettiin myös vaihtoehtoinen -B-lippu, jolla voidaan erikseen määrittää hakemisto, johon kääntöohjeet tallennetaan. Tähän laitettiin aiemmin luotu build-hakemisto. Ilman -B-lippua CMake tallentaa kääntöohjeet nykyiseen työkansioon. CMaken avatessa lähdekoodit ensimmäistä kertaa se on täysin tyhjä. Alkukonfigurointi voidaan tehdä painamalla c-näppäintä, jolloin CMake lukee lähdekoodeista tarvittavat kirjastot ja ohjelmistot, sekä konfiguraatioasetukset ja luo niiden perusteella kääntöohjeet.

Ensimmäisen konfiguroinnin jälkeen CMake näyttää listan kaikista asetuksista sekä niiden tiloista. (Kuvio 6)



Kuvio 6. ccmake ensimmäisen konfiguroinnin jälkeen

Asetuksia voidaan muuttaa enter-näppäimellä. Tärkein asetus, joka täytyy ensiksi muuttaa, on nimeltään *OPENCV_EXTRA_MODULES_PATH*. Tämän arvoksi tulee laittaa se hakemisto, johon ekstramoduulit ladattiin. Kun asetus on määritelty, voidaan tehdä uusi konfiguraatio painamalla c-näppäintä uudestaan. Nyt CMake käy läpi myös ekstramoduulit, ja näyttää niiden uudet konfiguraatioparametrit tähtimerkeillä. Seuraavaksi varmistetaan, että uuden asetuksen *BUILD_opencv_superres* arvo on. Tällöin CMake kääntää superres-moduulin, jota tarvitaan kuvien ylösskaalaamiseen. Seuraavaksi varmistetaan, että CUDA-tuki on päällä laittamalla asetuksen *WITH_CUDA* arvoksi *ON*. On myös hyvä idea tarkistaa, että CMake on varmasti löytänyt tarvittavat Python-kirjastot katsomalla, että *PYTHON3*-alkuisilla asetuksilla on oikeat arvot. Näin Python-moduuli voidaan kääntää, ja OpenCV:tä käyttää sen kautta. Myös asetuksen *OPENCV_ENABLE_NONFREE* arvoksi voidaan laittaa *ON*, mikäli halutaan, että myös esimerkiksi suljetun lähdekoodin algoritmit sisällytetään OpenCV:hen. Viimeisenä asetuksena kannattaa muuttaa asetuksen

CMAKE_INSTALL_PREFIX arvoa. Tämä asetus kertoo CMakelle, mihin hakemistoon valmis, käännetty ohjelmisto tulisi asentaa. Oletuksena tämä on */usr/local*. Tämä kuitenkin kannattaa muuttaa, sillä koska kyseistä ohjelmistoa ei ole asennettu paketinhallintatyökalun avulla, sen poistaminen voi olla hyvin vaikeaa, sillä kaikki CMaken asentamat tiedostot tulisi manuaalisesti poistaa. Siksi kannattaakin muuttaa *CMAKE_INSTALL_PREFIX*:in arvo johonkin uuteen, tyhjään hakemistoon, jolloin ohjelmiston poistaminen onnistuu yksinkertaisesti poistamalla kyseinen hakemisto *rm*-komennolla. Hyvä hakemisto olisi esimerkiksi */opt/opencv*.

Kun kaikki asetukset on säädetty, voidaan tehdä viimeinen konfigurointi. Tämän jälkeen kääntöohjeet voidaan luoda g-näppäintä painamalla. Kääntöohjeiden luonnin jälkeen ccmake sulkeutuu automaattisesti. Kääntäminen voidaan aloittaa vaihtamalla työhakemisto build-hakemistoon cd-komennolla, ja suorittamalla komento *make*. OpenCV on hyvin iso ja monimutkainen ohjelmisto, joten sen kääntämisessä voi mennä useita tunteja. Kun kääntäminen on vihdoinkin valmis, voidaan OpenCV asentaa suorittamalla komento *make install* pääkäyttäjän oikeuksilla. Tämä asentaa kaikki käännetyt tiedostot siihen hakemistoon, joka määriteltiin CMAKE_INSTALL_PREFIX:illa. Käännetyt tiedostot voidaan tämän jälkeen siivota build-hakemistosta komennolla *make clean*. Tämän jälkeen OpenCV on valmis käytettäväksi.

4.2 Nagios & Nginx

Nagios on avoimen lähdekoodin monitorointiohjelmisto, joka haluttiin asentaa analytiikkapalvelimelle sen varalle, että jotain hajoaa. Nagios monitoroi järjestelmän resursseja, ja lähettää ilmoituksia, mikäli jotain epänormaalia havaitaan. Nagios pystyttiin asentamaan apt:n kautta nimellä *nagios4*. Tämän peruspaketin lisäksi piti myös ladata *nagios4-cgi*-paketti, joka sisältää Nagiosin CGI-tiedostoja (Common Gateway Interface), joilla Nagiosia pystytään konfiguroimaan web-selaimen avulla. Näiden CGI-tiedostojen näyttämiseen tarvittiin myös HTTP-palvelin (Hypertext Transfer Protocol), johon valittiin Nginx. Nginx on HTTP-palvelin, joka erikoistuu nopeuteen, ja onkin voittanut suositun Apache-palvelimen testeissä,

joissa mitattiin HTTP-palvelimien suorituskykyä. (NGINX 2022) CGI-tiedostojen näyttämiseen tarvittiin HTTP-palvelimen lisäksi ylimääräinen fcgi-palvelin (FastCGI), jonka kautta CGI-tiedostot kävisivät ennen lähetystä käyttäjälle. Lopuksi tarvittiin vielä PHP-palvelin (PHP: Hypertext Preprocessor) Nagiosin PHP-tiedostojen käsittelyyn. Kaikki näistä pystyttiin asentamaan apt:n avulla komennolla *apt install nginx php8.0-fpm fcgiwrap*.

4.2.1 Nginx:in konfigurointi

Nginx:in konfigurointi aloitettiin luomalla uusi konfiguraatiotiedosto hakemistoon */etc/nginx/sites-available* nimeltä *nagios.conf*. Ensiksi luotiin peruskonfiguraatio, joka näyttää Nginx:in oletussivun oletusportissa 80. Konfiguraatioon lisättiin myös yksinkertainen salasanatunnistus *auth_basic* ja *auth_basic_user_file*-direktiiveillä. Tämän jälkeen luotiin erikseen salasanatunnistukseen käytettävä käyttäjätiedosto *htpasswd*-ohjelmalla, johon luotiin yksi nagios-niminen käyttäjä. Tämän jälkeen lisättiin location-direktiivillä uusi sijainti */nagios*-osoitteeseen. Direktiivin sisälle lisättiin ensin alias-komento hakemistoon */usr/share/nagios4/htdocs*. Tämä tarkoittaa, että kun käyttäjä menee www-selaimella polkuun /nagios, hakee Nginx hakemistosta */usr/share/nagios4/htdocs* index-tiedoston, jonka lähettää käyttäjälle. Viimeiseksi luotiin uusi location-direktiivi PHP-tiedostojen käsittelyä varten. Kun palvelimelta pyydetään PHP-tiedosto, lähettää Nginx sen ensin PHP-palvelimelle prosessoitavaksi *fastcgi_pass*-direktiivillä määrättyyn osoitteeseen. Tämän direktiivin arvoksi määriteltiin *unix:/run/php/php8.0-fpm.sock*, joka lähettää datan */run/php*-hakemistossa sijaitsevaan pistoketiedostoon (socket file). Pistoketiedostot ovat ohjelmistojen luomia erikoistiedostoja, joihin kirjoittamalla voidaan kommunikoida ohjelmiston kanssa. PHP-palvelin käsittelee PHP-koodin, ja lähettää prosessoidun HTML-datan (HyperText Markup Language) takaisin Nginx:ille, joka taas lähettää sen käyttäjälle.

Tämän jälkeen lisättiin vielä yksi location-direktiivi osoitteeseen */cgi-bin/nagios4* CGI-tiedostojen käsittelyyn. Tämä toimi samalla tavalla kuin edellisessä PHP-sijainnissa. Kun käyttäjä pyytää CGI-tiedostoa, hakee Nginx sen hakemistosta

/usr/lib/cgi-bin/nagios4. Tämän jälkeen Nginx lähettää CGI-tiedostot prosessointiin fcgi-palvelimelle pistoketiedostoa käyttäen, ja fcgi palauttaa prosessoidun HTML-datan Nginx:ille. Konfiguraatio oli nyt valmis, ja se voitiin ottaa käyttöön tekemällä symbolinen linkki. Symbolinen linkki on erikoistiedosto, joka osoittaa johonkin toiseen tiedostoon tai hakemistoon. Ne toimivat jokseenkin samalla tavalla kuin pikakuvakkeet Windows-käyttöjärjestelmässä (Ward 2021). Symbolisia linkkejä voidaan luoda *ln*-komennolla. Symbolinen linkki luotiin hakemistoon */etc/nginx/sites-enabled* nimellä *nagios.conf*, joka laitettiin osoittamaan tiedostoon *../sites-available/nagios.conf*. Tämä tarkoitti, että linkki osoitti hakemistossa */etc/nginx/sites-available* sijaitsevaan *nagios.conf*-tiedostoon. Konfiguraatio olisi voitu ottaa käyttöön myös yksinkertaisesti kopioimalla se *sites-enabled*-hakemistoon, mutta tällöin tiedosto olisi pitänyt kopioida uudestaan aina, kun sitä muokataan.

Viimeisenä asiana Nginx:in konfiguraatiot piti ladata uudestaan systemd:tä käyttäen. Ennen sitä oli kuitenkin hyvä tarkistaa konfiguraatiotiedosto syntaksivirheiden varalta, sillä muuten Nginx ei suostu käynnistymään. Testaaminen tehdään suorittamalla komento *nginx* lipulla *-t*. Nginx ilmoitti, että konfiguraatiossa ei ole virheitä, joten se voidaan ladata uudelleen. Tämä onnistui komennolla *systemctl reload nginx*. Nagiosin hallintapaneeliin pääsi nyt polusta */nagios*, mutta sivu ei näkynyt oikein. Tämä johtui siitä, että Nagiosin apt-paketti asensi sivuston eri osat ympäri tiedostojärjestelmää, jolloin esimerkiksi tarvitut CSS-tyylitiedostot (Cascading Style Sheets) olivat */usr/share/nagios4/htdocs*-hakemiston sijaan hakemistossa */etc/nagios4/stylesheets*. Tämä korjattiin luomalla jokaiselle puuttuvalle asialle oma location-direktiivinsä, johon määriteltiin alias-direktiivillä hakemisto, josta kyseiset tiedostot löytyvät. Salasanakirjautumista muutettiin myös siten, että se oli päällä vain */nagios*-polussa, sillä muuten koko sivuston käyttämiseen pitäisi tunnistautua, joka haittaisi muiden http-sovellusten käyttämistä. Tämän jälkeen sivusto latautui oikein ilman virheitä.

4.2.2 Nagiosin konfigurointi

Nagiosissa on neljäntyyppisiä konfiguraatioita: pääkonfiguraatiotiedosto, joka määrittää direktiivejä, jotka vaikuttavat Nagiosin toimintaan, resurssitiedostot, joihin voidaan tallentaa käyttäjän omia tietoja kuten salasanoja, objektitiedostot, joissa määritellään erinäisiä Nagiosin objekteja, kuten palveluita ja komentoja, ja CGI-konfiguraatiotiedosto, joka määrittää, miten CGI-tiedostot toimivat. (Nagios 2022) Konfigurointi aloitettiin CGI-konfiguraatiotiedostosta, jotta CGI-tiedostot saataisiin toimimaan oikein. CGI-konfiguraatiotiedosto sijaitsi */etc/nagios4/cgi.cfg*-tiedostossa, josta vaihdettiin asetuksen *use_authentication* arvoksi 1. Näin CGI-tiedostot käyttävät käyttäjätunnistautumista, jolloin vain tietyt käyttäjät voivat hallinnoida Nagiosia CGI-tiedostojen kautta. Nginx välittää salasanatunnistuksella kirjautuneen käyttäjän nimen Nagiosille, jolloin Nagios osaa määrittää oikeat käyttöoikeudet. Tämän jälkeen nagios-käyttäjä lisättiin direktiiveihin *authotized_for_system_information*, *authorized_for_configuration_information*, *authorized_for_system_commands*, *authorized_for_all_services*, *authorized_for_all_hosts*, *authorized_for_all_service_commands* ja *authorized_for_all_host_commands*. Tämä antoi nagios-käyttäjälle kaikki mahdolliset oikeudet muokata Nagiosin asetuksia CGI-tiedostojen kautta. Tämä tehtiin väliaikaisesti siksi aikaa, kun Nagiosia konfiguroitiin, sillä nagios-käyttäjän jättäminen tällaiseksi olisi tietoturvariski, mikäli Nagiosiin sallittaisiin pääsy ulkoverkon kautta.

4.2.3 Monitoroitavien palveluiden määrittely

Seuraavaksi asennettiin kaikki Nagiosilla monitoroitavat palvelut. Nagiosilla haluttiin monitoroida järjestelmän suoritinkäyttöä, vapaan muistin ja levytilan määrää, HTTP- ja SSH-palvelimien toimivuutta sekä CUDA:n toimivuutta. Näistä kaikki muut paitsi CUDA olivat jo valmiiksi luotuja, joten ne eivät vaatineet ylimääräistä konfigurointia. CUDA:a varten piti luoda uusi objektitiedosto, jossa määriteltiin uusi komento ja palvelu. Nagiosin objektitiedostot sijaitsivat hakemistossa */etc/nagios-plugins/config*, johon luotiin uusi *cuda.cfg*-niminen tiedosto. Objektitiedostoissa objektit määritellään define-sanalla, jonka jälkeen määritellään, millainen objekti

ollaan luomassa. Ensiksi luotiin command-objekti, joka on suoritettava komento. Sen nimeksi laitettiin *check_cuda*, ja suoritettavaksi komennoksi skripti */usr/bin/nagios_check_cuda.sh*. Tämän jälkeen cuda.cfg-tiedosto tallennettiin, ja kyseinen skripti luotiin.

Skripti toteutettiin bash-komentotulkin omilla skriptausominaisuuksilla. Skriptissä ensin luotiin muuttuja *cuda_version*, jonka arvoksi tuli komennon *nvidia-smi | grep "CUDA Version" | awk '{print $9}'* lopputulos. *nvidia-smi* on komento, joka listaa Nvidian näytönohjaimen ja sen ajureiden tiedot. Tämän komennon tulos putkitettiin grep-komennolle, jolla voidaan suodattaa tekstiä. Putkituksella voidaan uudelleenohjata jonkin komennon tulos jollekin toiselle komennolle. Sitä merkitään |-merkillä. grep siis suodattaa pois kaikki rivit, joilla ei ole "CUDA Version"-tekstiä. Lopuksi tämä vielä putkitettiin *awk*-komennolle, jolla voidaan suodattaa tekstiä kokeneemmin kuin grep:llä. *print $9* kertoo awk:lle suodattaa pois kaiken tekstin paitsi yhdeksännen sanan, joka on tässä kyseisessä tekstissä "CUDA Version"-tekstin jälkeen tuleva CUDA:n versionumero. Tämän jälkeen muuttujan arvoa verrataan if-lausekkeella. Jos muuttujan arvo on erisuuri kuin "N/A", tulostaa skripti CUDA:n version ja sulkeutuu koodilla 0. Tämä kertoo Nagiosille, että palvelu toimii oikein ja että siinä ei ole virheitä. Jos taas arvo on "N/A", tulostaa skripti, että CUDA:n versiota ei voitu tarkistaa, ja sulkeutuu koodilla 2. Tämä taas kertoo Nagiosille, että palvelussa on vakava virhe. Lopuksi vielä skriptille lisättiin suoritusoikeudet *chmod +x /usr/bin/nagios_check_cuda.sh*-komennolla.

Tämän jälkeen palattiin muokkaamaan */etc/nagios-plugins/config/cuda.cfg*-tiedostoa, johon luotiin uusi palveluobjekti, joka on monitoroitava palvelu. Palvelun direktiivin *host_name* arvoksi annettiin *localhost*, jolloin tämä palvelu toimii vain paikallisella palvelimella. Direktiivin *check_command* arvoksi annettiin *check_cuda*, joka viittaa aikaisemmin luotuun komentoon. Näin aina kun palvelun tila tarkistetaan, suorittaa Nagios komennon *check_cuda*. Direktiivin check_period arvoksi laitettiin vielä 24x7, jolloin palvelun tila tarkistetaan 24 kertaa seitsemänä

päivänä viikossa, ja direktiivin *contacts* arvoksi *slack-notifications*, jota käytettäisiin myöhemmin Slack-viestien lähetykseen.

Slack-viestien lähetykseen kirjoitettiin Python-skripti, joka ottaa argumentteina ilmoituksen tiedot, rakentaa niistä JSON-vastauksen sekä lähettää sen Slackin palvelimelle, jossa se ohjautuu Ajapaikin Slack-kanavalle. Tämän jälkeen luotiin uusi Nagiosin objektitiedosto, johon määriteltiin uusi komento nimeltä *slack-notification*, joka suorittaa aiemmin luodun Python-skriptin parametreilla, jotka Nagios täyttää ilmoituksen tiedoilla. Tiedostoon luotiin myös uusi kontaktiobjekti, jossa määriteltiin, kuinka usein ilmoituksia lähetetään sekä komennon, joka suoritetaan, kun ilmoitus lähetetään. Täksi komennoksi laitettiin aiemmin luotu *slack-notification*-komento. Nyt Slack-viestit lähetetään kaikista niistä palveluista, joiden kontaktiksi on määritelty *slack-notifications*. Lopuksi vielä määriteltiin tämä uusi kontakti kaikkiin Nagiosin monitoroimiin palveluihin, jolloin ilmoituksia lähetettäisiin kaikista näistä palveluista.

Kun oli varmistettu, että kaikki toimii, muutettiin vielä Nagiosin CGI-asetuksia niin, että nagios-käyttäjä poistettiin asetuksista authorized_for_system_commands, authorized_for_configuration_information, authorized_for_all_service_commands ja authorized_for_all_host_commands. Näin nagios-käyttäjällä pystyttiin vain monitoroimaan dataa. Nyt Nagiosiin voitiin sallia pääsy ulkoverkosta ilman tietoturvariskiä.

4.3 SSH-tunnelointi ja SSHFS

SSH-protokollaa voidaan etäyhteyden lisäksi käyttää myös yhteyksien tunnelointiin ja tiedostonsiirtoon. SSH-tunneloinnilla kaksi tietokonetta voi siirtää tiettyyn porttiin tulevaa liikennettä keskenään. Näin esimerkiksi etätietokoneen portti 80 voidaan uudelleenohjata paikallisen koneen porttiin 8080, jolloin pystytään kommunikoimaan etäkoneen http-palvelimen kanssa portista 8080. Tunnelointi toteutetaan ssh-komennon lipuilla -L ja -R. -L tarkoittaa paikallista tunnelointia, jolloin

kaikki paikalliseen porttiin tuleva liikenne ohjataan etäporttiin. -R taas on etätunnelointi, jossa etäpalvelimelta voidaan ohjata portti paikalliseen palvelimeen. Esimerkiksi komento *ssh -L 8080:localhost:80 user@palvelin.com* yhdistäisi käyttäjänä user palvelimeen palvelin.com, ja tunneloisi paikalliseen porttiin numero 8080 tulevat yhteydet etäporttiin numero 80.

Yleisin käyttötapa tälle analytiikkapalvelimella on ohjata sen portteja käyttäjien tietokoneille. Näin esimerkiksi analytiikkapalvelimen tietokantaa voidaan käyttää suoraan käyttäjän omalta koneelta. Ajapaikin pääpalvelimella tunnelointia haluttiin käyttää siten, että Ajapaikin pääpalvelimen porttiin 8022 yhdistämällä liikenne ohjautuisi analytiikkapalvelimen porttiin 22. Samanlainen tunnelointi haluttiin myös http-yhteyksille porttiin 8080. Näin muun muassa analytiikkapalvelimen Nagios-hallintapaneelia voitaisiin käyttää yhdistämällä Ajapaikin pääpalvelimen porttiin 8080. Tämä toteutettiin lisäämällä Ajapaikin pääpalvelimelle järjestelmän käynnistyessä suoritettava skripti, joka yhdistää analytiikkapalvelimelle ssh-komennolla, jolle on annettu paikallisen tunneloinnin lippu -L. Komento pääpalvelimen portin 8022 ohjaamiseksi analytiikkapalvelin porttiin 22 oli siis *ssh -L 8022:localhost:22 käyttäjä@analytiikkapalvelin*. Tämän perään lisättiin vielä toinen -L-lippu, jonka arvoksi laitettiin *8080:localhost:80*. Näin myös portti 8080 ohjautuu porttiin 80.

Tiedostojen siirtoa SSH-protokollan kautta kutsutaan yleisesti SFTP:ksi (Secure File Transfer Protocol). Tiedostojen siirto toteutettiin käyttäen SSHFS:ää, joka on FUSE-pohjainen tiedostojärjestelmä (Filesystem in Userspace). SSHFS:än avulla voidaan hakemistoon liittää SFTP-yhteys, jonka kautta etätiedostoja voidaan kopioida, siirtää ja poistaa niin kuin ne olisivat paikallisella koneella. (ArchWiki 2022) Koska SSHFS on FUSE-pohjainen, ei sen käyttämiseen tarvita pääkäyttäjän oikeuksia. Yhteys toteutettiin Ajapaikin pääpalvelimen ja analytiikkapalvelimen välille, jolloin esimerkiksi varakopioita ja muita tiedostoja voidaan siirtää palvelimien välillä yksinkertaisesti.

Prosessi aloitettiin luomalla sekä pääpalvelimelle että analytiikkapalvelimelle uudet käyttäjät, joidenka kautta SSHFS:ää käytettäisiin. Pääpalvelimen käyttäjälle luotiin myös avaintiedosto, jonka avulla tunnistautuminen tehtäisiin. Tämän jälkeen testattiin, toimiiko SSHFS manuaalisesti vaihtamalla uudelle käyttäjälle *sudo -u*-komennolla ja suorittamalla sshfs-komento. sshfs:än käyttö on samanlaista kuin ssh-komennon käyttö, ja määrittelemällä *-i*-lipun pystyttiin käyttämään luotua avaintiedostoa tunnistautumiseen. Liitospisteenä käytettiin käyttäjän kotihakemistoon luotua *testi*-hakemistoa. SSHFS yhdisti onnistuneesti, ja nyt hakemisto oli suora linkki Ajapaikin pääpalvelimella sijaitsevan käyttäjän kotihakemistoon. Tätä testattiin kopioimalla hakemistoon tiedosto, ja tarkastamalla, löytyikö se pääpalvelimella kyseisen käyttäjän kotihakemistosta. Kun toimivuus oli testattu, liitos poistettiin komennolla *fusermount3 -u testi*.

Tämän jälkeen aloitettiin konfiguroimaan automaattista liitosta, jolla SSHFS saataisiin päälle aina järjestelmän käynnistyessä. Tämä tehtiin muokkaamalla /etc/fstab-tiedostoa, joka sisältää tietoa eri osioista ja laitteista, joita init-järjestelmä yrittää liittää järjestelmän käynnistyessä. fstab-tiedostossa on jokaisella rivillä kuusi saraketta, jotka tulee olla täytetty. Näistä ensimmäinen on laitetieto, joka määrittelee liitettävän laitteen tai tiedostojärjestelmän. Toinen sarake on hakemisto, johon laite liitetään. Kolmas on tiedostojärjestelmä, joka laitteella on. Tähän sarakkeeseen voidaan laittaa arvoksi myös auto, jolloin init-järjestelmä yrittää automaattisesti päätellä, mikä tiedostojärjestelmä on käytössä. Tämä on hyödyllistä esimerkiksi optisen median, kuten CD- ja DVD-asemien liittämisessä. Neljäs sarake on asetukset, jossa voidaan määritellä liitoksen tiedostojärjestelmäpohjaisia asetuksia. Viides sarake kertoo, tarkastetaanko liitos dump-ohjelmalla. Tämän arvoksi laitetaan yleensä 0, jolloin tarkistusta ei tehdä. Kuudes ja viimeinen sarake kertoo, tarkastetaanko liitoksen tiedostojärjestelmän kunto *fsck*-ohjelmalla. Tämän arvo tulisi olla juurihakemiston liitoksen kanssa 1, ja kaikkien muiden liitosten kohdalla 2. Arvoksi voidaan antaa myös 0, jolloin tarkistusta ei tehdä. (ArchWiki 2022)

/etc/fstab-tiedostoon lisättiin uusi rivi, jonka laitetiedoston sarakkeeseen laitettiin arvoksi pääpalvelimen SSH-osoite. Liitoksen hakemistoon laitettiin /mnt/ajapaik.ee-hakemisto. Tiedostojärjestelmäksi määriteltiin fuse.sshfs, ja asetuksiksi tarvittavat asetukset, joita tarvitaan SSHFS:än liittämiseen. Näistä tärkeimpiä olivat noauto, x-systemd.automount, _netdev, allow_other, default_permissions, uid ja gid. noauto tarkoittaa, että järjestelmä ei yritä liittää laitetta automaattisesti järjestelmän käynnistyessä. x-systemd.automount tarkoittaa, että systemd osaa automaattisesti liittää laitteen, kun /mnt/ajapaik.ee-hakemistoa käsitellään. _netdev kertoo, että kyseessä on verkkolaite. allow_other tarkoittaa, että myös muut käyttäjät kuin pääkäyttäjä voivat lukea ja muokata SSHFS:än kautta tiedostoja. default_permissions tarkoittaa, että SSHFS pyrkii säätämään paikalliset tiedostojen oikeudet samoiksi kuin mitä ne ovat etäpalvelimella. uid ja gid määrittävät käyttäjän ja ryhmän ID-tunnukset, joilla liitos liitetään. Dumpin ja fsck:in arvoiksi laitettiin nollat, sillä niiden käyttö ei ollut tarpeellista SSHFS:än kanssa sen FUSE-pohjaisuuden vuoksi. Tämän jälkeen systemd:n konfiguraatio ladattiin uudelleen *systemctl reload-daemon*-komennolla, jonka jälkeen liitos oli valmis käytettäväksi.

Myös käyttäjät voivat käyttää SSHFS:ää jakaakseen palvelimen levypinnan omille tietokoneilleen. Tämä onnistuisi Linux-järjestelmissä ensin asentamalla SSHFS jakelun pakettivarastoista, ja suorittamalla komento *sshfs käyttäjä@palvelin:~/hakemisto liitospiste*, jolloin SSHFS kirjautuisi palvelimelle *palvelin* käyttäjänä *käyttäjä*, ja liittäisi sieltä hakemiston nimeltä *hakemisto* käyttäjän kotihakemistosta paikalliseen hakemistoon nimeltä *liitospiste*. Tiedostoja voidaan siirtää myös SFTP- ja SCP-protokollilla. SCP (Secure Copy) on yksinkertaisempi versio SFTP:stä, ja se ei tue muun muassa etätiedostojen poistoa tai keskeytyneiden latausten jatkamista. Tiedostoja voidaan siirtää ja ladata käyttämällä joko *scp*- tai *sftp*-komentoa. scp-komennolle annetaan parametreiksi ensin tiedoston lähde, ja toiseksi päämäärä. Jos siis haluttaisiin ladata etäpalvelimelta tiedosto, komento olisi *scp käyttäjänimi@palvelin:tiedosto.txt /paikallinen/hakemisto/*. Jos taas haluttaisiin ladata palvelimelle tiedosto, suoritettaisiin komento *scp tiedosto.txt käyttäjänimi@palvelin:/etä/hakemisto/.* (ArchWiki 2022)

sftp-komennolle annetaan parametriksi yhdistettävä käyttäjänimi ja palvelimen osoite. Kirjautumisen jälkeen voidaan tiedostoja ladata palvelimelta *get*-komennolla, jolle annetaan ensimmäiseksi parametriksi halutun tiedoston nimi, ja toiseksi paikallisen hakemiston nimi, johon tiedosto ladataan. Tiedostoja voidaan taas ladata palvelimelle *put*-komennolla, jolle annetaan samalla tavalla parametreiksi ensin paikallisen tiedoston nimi ja sen jälkeen etähakemiston nimi.

4.4 Median ja tietokannan peilaaminen ja puhdistus

Analytiikkapalvelimelle haluttiin peilata Ajapaikin pääpalvelimen mediatiedostot sekä puhdistettu tietokanta. Tietokannan puhdistus tarkoittaa, että siitä on poistettu tai sensuroitu kaikki salassa pidettävät tiedot, kuten esimerkiksi käyttäjien salasanat ja henkilötiedot. Mediatiedostojen peilaaminen toteutettiin yksinkertaisesti rsync-ohjelmalla. rsync on ohjelma, jonka avulla voidaan peilata tiedostoja ja hakemistoja. rsync:iä voidaan käyttää sekä paikallisesti että etänä. rsync lataa etänä tiedostoja SSH-protokollan avulla. rsync-ohjelmalla peilattiin myös uusin tietokannan varmuuskopio analytiikkapalvelimelle.

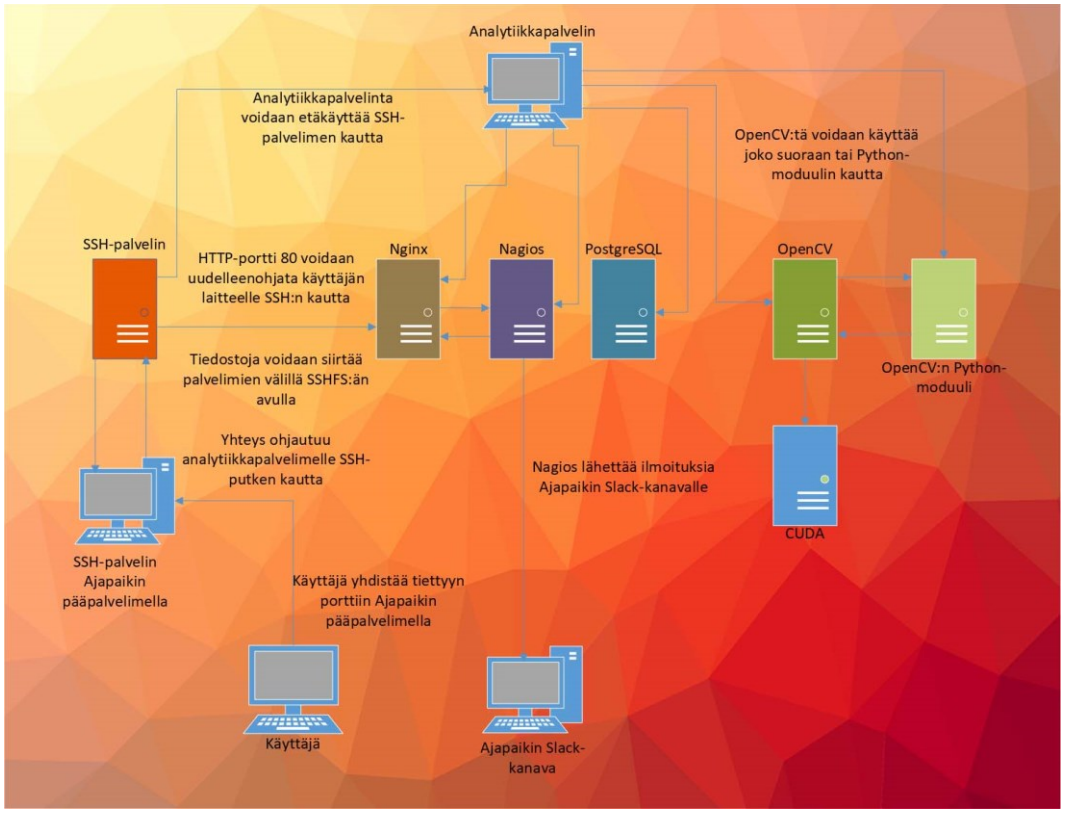
Tietokantaa varten palvelimelle asennettiin PostgreSQL-palvelin, joka on sama, mitä Ajapaik käyttää pääpalvelimellaan. Tietokantojen varmuuskopiot olivat salakirjoitettuja PGP:llä (Pretty Good Privacy), joka on avoimen lähdekoodin salausohjelma, joka tukee julkisen avaimen salausta. Salaukseen ja sen purkamiseen käytettiin GPG-ohjelmaa (GNU Privacy Guard). Salauksen purkamisen jälkeen tietokanta voitiin palauttaa paikalliselle PostgreSQL-palvelimelle. Ensiksi kuitenkin luotiin SQL-palvelimelle uusi tietokanta, johon varakopio palautettaisiin. Paikalliseen palvelimeen voitiin yhdistää *psql*-ohjelmalla, jolle annettiin lipuiksi *-h 127.0.0.1*, joka määritti, että yhdistettävä palvelin on paikallinen, *-p 5433*, joka määritti yhdistettäväksi portiksi portin 5433, sekä *-U postgres*, joka määritti yhdistäväksi käyttäjäksi postgres-nimisen käyttäjän. Tämän jälkeen ohjelma kysyi postgres-käyttäjän salasanaa, jonka jälkeen se kirjautui palvelimelle. Palvelimelle kirjauduttua uusi tietokanta voitiin tehdä *CREATE DATABASE*-käskyllä. Palvelimelle luotiin myös kaksi uutta käyttäjää, joista yhdellä oli täydet oikeudet ja toisella vain lukuoikeudet

uuteen tietokantaan. Tämän jälkeen palvelimelta kirjauduttiin ulos, ja varakopio palautettiin. Tämä onnistui *pg_restore*-ohjelmalla, joka tuli PostgreSQL:än apt-paketin mukana. Sille annettiin lipuiksi samat tiedot kuin aikaisemmalla psql-komennolla, sekä lisäksi -d-lippu, jolla määriteltiin tietokannan nimi, johon varakopio palautetaan. Tähän laitettiin äsken luotu tietokanta.

Palauttamisen jälkeen paikallisella SQL-palvelimella oli identtinen tietokanta kuin Ajapaikin pääpalvelimella. Tämän jälkeen käytiin läpi kaikki ne tietokannan taulut, joissa oli herkkäluontoisia tietoja. Näistä osa tyhjennettiin SQL-kielen *TRUNCATE*-käskyllä, joka tyhjentää kaiken datan taulun sisällä, muttei poista taulua kokonaan. Joissain tiedoissa jokaisen tietueen sarake korvattiin tyhjällä arvolla *UPDATE*-käskyllä. Joissain tapauksissa, joissa arvo ei voinut olla tyhjä, kuten käyttäjänimissä, laskettiin ensin alkuperäisestä arvosta MD5-tarkistussumma md5-funktiolla. Tämän jälkeen käytettiin left-funktiota, jolle annettiin arvoksi MD5-tarkistussumma sekä 16. Tämä tarkoittaa, että MD5-tarkistussummasta päivittyi tietueeseen vain sen ensimmäiset 16 kirjainta. Näin tarkistussumma ei ole täydellinen, eikä sitä enää voida palauttaa alkuperäiseen muotoon. Sähköposteissa käytettiin samaa tapaa, mutta niihin oli myös lisättävä *concat*-funktio, joka liittää yhteen kaksi tekstijonoa. Ensimmäiseksi arvoksi laitettiin sama 16-merkkinen MD5-tarkistussumma, ja toiseksi sähköpostiosoitteeseen kuuluva @email.com-pääte. Näin ohjelma ei valita epäkelvoista sähköposteista. Tietokannan puhdistuksen jälkeen tehtiin vielä bash-skripti, joka automatisoisi prosessin. Tätä ei kuitenkaan laitettu suorittumaan säännöllisesti, sillä analytiikkapalvelimen käytön kannalta ei tarvittu ajantasaista versiota tietokannasta. Tämä myös vaatisi salakirjoituksen purkamiseen tarvittavien avaimien säilyttämisen palvelimella, mikä ei olisi tietoturvallista, sillä analytiikkapalvelimelle annetaan käyttäjille pääkäyttäjän oikeudet kevyemmin ohjelmistokehitystä varten. Tämän takia ulkopuoliset voisivat päästä salakirjotusavaimiin käsiksi.

4.5 Tulokset

Lopputuloksena projektissa on etäkäytettävä palvelin, jolla harrastelijat ja opiskelijat voivat tehdä koneoppimiseen liittyviä projekteja. Palvelimelle asennettiin CUDA ja OpenCV, jotka mahdollistavat koneoppimiseen liittyvien tehtävien tekemisen. Analytiikkapalvelinta on jo käytetty onnistuneesti Tarton yliopiston puolelta, jonka oppilaat käyttivät sitä koneoppimiskurssillaan kuvien alueiden tunnistamiseen. Tämän avulla valokuvista voidaan tunnistaa, mikä osa siitä on varsinaista kuvaa ja mikä osa kehystä ja muuta taustaa. Tämä toimi hyvin, ja kone pystyi tunnistamaan kuvan keskimäärin 98 prosentin tarkkuudella. Yliopiston opiskelijat olivat tyytyväisiä analytiikkapalvelimen toimintaan. Palvelimelle asennettu ja konfiguroitu Nagios myös jatkuvasti monitoroi palveluita ja ilmoittaa, mikäli johonkin niistä tulee ongelma. Näin palvelinta ei tarvitse jatkuvasti itse monitoroida ongelmatilanteiden varalta, ja Python-skriptin lähettämien Slack-viestien ansiosta niihin voidaan reagoida nopeasti. SSH-tunneloinnin avulla analytiikkapalvelimeen voidaan yhdistää suoraan Ajapaikin pääpalvelimen kautta, jolloin se ei tarvitse julkista tai staattista IP-osoitetta. Tämä myös parantaa tietoturvaa, sillä yhdistävät osapuolet eivät ole suoraan yhteydessä analytiikkapalvelimeen. Konfiguroidun SSHFS:än avulla voidaan helposti siirtää tiedostoja pääpalvelimen ja analytiikkapalvelimen välillä. SFTP:n avulla voidaan toteuttaa helposti esimerkiksi varmuuskopiointi, joka voidaan ajastaa tekemään varmuuskopiot ja lataamaan ne SFTP:n avulla etäpalvelimelle. Median ja tietokannan peilaaminen analytiikkapalvelimelle mahdollistavat Ajapaikin kehityksen. Näin voidaan testata uusia muutoksia Ajapaikiin ensin analytiikkapalvelimella ja katsoa, miten se toimii ennen sen siirtämistä produktiokäyttöön. Ohessa on vielä kuvio, jossa on kuvailtu järjestelmän rakenne kaaviona. (Kuvio 7)



Kuvio 7. Järjestelmän rakenne

5 YHTEENVETO JA POHDINTA

Tässä opinnäytetyössä olen käsitellyt Linux-palvelimen asennusta ja konfigurointia sekä kuvaillut analytiikkapalvelimen asennusta vaihe vaiheelta. Kuvailin Linux-palvelimen asennusta yleisesti alkaen Linux-jakelun valinnasta ja komentojonon käytöstä aina palvelinkäytössä tärkeiden SSH-palvelimen sekä palomuurin konfigurointiin. Tätä kappaletta kirjoittaessani tajusin, että Linux-järjestelmä on valtava kokonaisuus, jota ei mitenkään saa mahdutettua yhteen opinnäytetyöhön. Koitin kirjoittaa tekstini niin, että sitä ymmärtäisi myös Linuxista mitään tietämätön henkilö, mutta huomasin, että siltikään tekstistä ei välttämättä ymmärrä joitain asioita, mikäli henkilöllä ei ole minkäänlaista kokemusta Linuxista. Muuten koen onnistuneeni Linux-palvelimen asennuksen kuvailemisessa, ja onnistuin kuvailemaan kaikki ne tärkeimmät seikat, jotka tulee ottaa huomioon palvelinkäytössä.

Analytiikkapalvelimen asennuksesta kerroin vaihe vaiheelta, miten analytiikkapalvelin asennettiin sekä kuvailin tehtyjä konfiguraatioita. Näistä varmasti isoin oli Nginx:in ja Nagiosin asennus, jonka asennuksen kuvaamisessa jouduin käyttämään paljon erilaista termistöä. En myöskään kerennyt hirveämmin perehtymään kaikkiin niihin ominaisuuksiin, joita Nagios tarjoaa, vaan raapaisin vain pintaa luomallani CUDA-tarkistusskriptillä. Tässä voisikin ehkä olla aihe tulevaan opinnäytetyöhön. CUDA:n ja OpenCV:n asennuksessa minulla meni alun perin paljon aikaa, sillä oikeiden konfiguraatioasetusten sekä tarvittavien kirjastojen etsimisessä kesti, ja ohjelma piti kääntää useaan otteeseen uudestaan. En myöskään pystynyt testaamaan OpenCV:n toimivuutta omalla tietokoneellani, sillä näytönohjaimessani ei ollut tarpeeksi muistia ylösskaalausta varten. SSH-tunneloinnissa ja SSHFS:än asennuksessa minulla ei ollut suurempia ongelmia. Koen, että analytiikkapalvelimen asennus onnistui, ja että sitä voidaan käyttää vielä monissa, koneoppimiseen liittyvissä tehtävissä sekä kehitystyössä. Pysyin myös suurimmalta osin aikataulussa asennukseen liittyvissä tehtävissä.

Loppujen lopuksi opinnäytetyöstä voidaan päätellä, että Linux on kevyt, turvallinen ja loppupeleissä helppokäyttöinen järjestelmä palvelinkäyttöön, joka sopii erinomaisesti sekä simppeleiden että vaativien palvelimien pyörittämiseen. Opinnäytetyön tavoitteet täyttyivät, ja pystyin suurimmilta osin kuvailemaan Linux-järjestelmän toimintaa sekä palvelinkäytössä huomioon otettavia seikkoja. Keskeiseen tutkimuskysymykseen, eli Linux-järjestelmän toimintaan, siis vastattiin myös. Opinnäytetyössä helpoimmat osuudet tehdä olivat Linux-järjestelmän yleiseen käyttöön liittyvät, kuten komentojonon käyttö ja paketinhallinta, sillä nämä ovat tulleet itselleni hyvin tutuiksi omien Linux-palvelimieni konfiguroinnin pohjalta. Vaikein oli init-järjestelmäkappaleen tekeminen, sillä systemd on valtavan iso ohjelma, johon sisältyy monia eri osa-alueita, ja en kyseisessä kappaleessa kerennyt kertomaan siitä ja sen mahdollisista ominaisuuksista kuin hyvin vähän. Uskon, että systemd on niin iso kokonaisuus, että pelkästään siitä saisi tehtyä oman opinnäytetyönsä.

Toivon, että tästä opinnäytetyöstä olisi hyötyä palvelinta käyttäville opiskelijoille heidän ohjelmointiprojekteissaan sekä Ajapaikin koneoppimisprojekteissa. Toivoisin myös, että tästä olisi hyötyä sellaisille henkilöille, jotka ovat kiinnostuneita Linuxista sekä sen palvelinkäytöstä. Parhain tapa oppia Linuxin toimintaa on minulla itselläni ollut asioiden tekeminen ja kokeileminen sekä tietysti epäonnistuminen. Suosittelisinkin sellaisille henkilöille, jotka haluavat oppia lisää, katsomaan esimerkiksi DigitalOcean-sivuston oppaita, josta löytyy hyvin paljon tietoa sekä Linuxin käytöstä että erinäisten palvelinohjelmistojen konfiguroinnista ja harjoittelemaan esimerkiksi virtuaalikoneella näiden asennusta. Itse olen tämän opinnäytetyön tekemisen aikana pystynyt syventämään tietojani Linuxista, sekä päässyt asentamaan ja konfiguroimaan sellaisia asioita, johon minulla ei ole ennen ollut mahdollisuutta. Koenkin, että olen hyötynyt opinnäytetyön teosta, ja pystyn varmasti hyödyntämään näitä tietoja tulevaisuudessa työelämässä.

LÄHTEET

Arch Linux. 2022. fstab. Viitattu 23.2.2022. https://wiki.archlinux.org/title/Fstab

Arch Linux. 2022. SCP and SFTP. Viitattu 7.4.2022. https://wiki.archlinux.org/title/SCP_and_SFTP

Arch Linux. 2022. SSHFS. Viitattu 21.2.2022. https://wiki.archlinux.org/title/SSHFS

CMake. 2022. Overview. Viitattu 5.2.2022. https://cmake.org/overview/

Debian Project. 2021. Reasons to use Debian. Viitattu 22.12.2021. https://www.debian.org/intro/why_debian

Flaunt Digital. 2018. Why You Should Use Linux For Your Servers. Viitattu 29.3.2022. https://flauntdigital.com/blog/why-use-linux-servers/

Linux.fi. 2022. Tiedoston oikeudet. Viitattu 15.1.2022. https://www.linux.fi/wiki/Tiedoston_oikeudet

Linux manual. 2022. Description of the filesystem hierarchy. Viitattu 21.12.2021. https://man7.org/linux/man-pages/man7/hier.7.html

Nagios Enterprises. 2022. Configuration Overview, Nagios Core Documentation. Viitattu 14.2.2022. https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/config.html

Nagios Enterprises. 2021. Nagios - The Industry Standard In IT Infrastructure Monitoring. Viitattu 11.10.2021. https://www.nagios.org/

Negus, C. 2015. Linux Bible. 9. painos. Indianapolis. John Wiley & Sons, Inc.

NGINX. 2022. What is NGINX? Viitattu 9.2.2022. https://www.nginx.com/resources/glossary/nginx/

OpenCV. 2021. About OpenCV. Viitattu 21.12.2021. https://opencv.org/about/

phoenixNAP. 2021. Linux vs. Windows Server: The Ultimate Comparison. Viitattu 29.3.2022. https://phoenixnap.com/blog/linux-vs-microsoft-windows-servers

The PostgreSQL Global Development Group. PostgreSQL: The world's most advanced open source database. Viitattu 11.10.2021. https://www.postgresql.org/

Quest. 2022. What is Active Directory? How does it work? Viitattu 29.3.2022. https://www.quest.com/solutions/active-directory/what-is-active-directory.aspx

Rothman, M. 2010. Network Security Fundamentals: Default Deny. Viitattu 24.1.2022. https://securosis.com/blog/network-security-fundamentals-default-deny

SSH.COM. 2022. What is SSH Public Key authentication? Viitattu 22.1.2022. https://www.ssh.com/academy/ssh/public-key-authentication

Ubuntu. 2022. Package Management. Viitattu 6.1.2022. https://ubuntu.com/server/docs/package-management

Ubuntu. 2022. UFW - Community Help Wiki. Viitattu 24.1.2022. https://help.ubuntu.com/community/UFW

W3Techs. 2021. Usage statistics of operating systems for websites. Viitattu 21.12.2021. https://w3techs.com/technologies/overview/operating_system

Ward, B. 2021. How Linux Works: What Every Superuser Should Know. 3. painos. San Francisco. No Starch Press.

Wiki.GIS, 2022. PostGIS. Viitattu 7.4.2022 http://www.wiki.gis.com/wiki/index.php/PostGIS